TRABAJO FIN DE GRADO

Grado en Ingeniería de Sistemas Audiovisuales

DESARROLLO DE UNA APLICACIÓN EN SEGUNDA PANTALLA PARA GOOGLE TV

Autor: Diego Martínez Tejero
Tutor: Ángel García Crespo

Universidad
Carlos III de Madrid

TÍTULO: ***DESARROLLO DE UNA APLICACIÓN EN SEGUNDA PANTALLA PARA GOOGLE TV***

AUTOR: ***DIEGO MARTÍNEZ TEJERO***

TUTOR: ***ÁNGEL GARCÍA CRESPO***

FECHA: ***septiembre 2013***

“Trata de no considerar inteligentes sólo a quienes piensan como tú”
Hugo Ojetti

Agradecimientos

Termina aquí una etapa de mi vida, siendo el último capítulo la entrega y presentación de este trabajo fin de grado.

Cuando empecé la carrera, qué lejano quedaba este día, pero al final, todo llega. Muchas han sido las circunstancias que han rodeado mi periplo por la Universidad y me gustaría acordarme de varias personas que han estado ahí estos años en estas pequeñas líneas antes de meterme de lleno en el desarrollo de la memoria.

No quiero centrarme sólo en el último periodo llevando a cabo este proyecto, sino en toda mi carrera, que ha estado marcada por las personas que me han rodeado y es a ellas a quienes, desde aquí, voy a agradecer el haber estado conmigo en muchos momentos.

Gracias a mi familia. Mis padres y mis dos hermanas, son para mí un pilar en el que apoyarme día tras día y gracias a ellos he conseguido ser mejor persona y conseguir mis objetivos. Desde luego, gran parte de todo lo que he conseguido es gracias a su cariño y empuje.

Gracias a mi novia, María. Ella ha tenido que soportarme en muchos momentos de malhumor, de frustración en épocas de exámenes, en periodos de estrés, y en todos los casos me ha sacado una sonrisa, me ha comprendido y me ha ayudado. Gracias Mary.

Gracias a mis amigos, tanto de la carrera como los de toda la vida. María, Andrea… qué buenos ratos de biblioteca hemos pasado. Pero en especial gracias a mis tres compañeros de fatigas, Izan, Miguel y Dani. Son tres personas muy importantes en mi vida, incluso, quién sabe, si no hubiera tenido la ayuda de Miguel en muchas ocasiones ahora quizás no estaría redactando estas líneas.

No puedo saltarme a mis compañeros de trabajo de Optiva Media, en especial a Mario y Natalia que me han ayudado mucho durante todo el trabajo fin de grado. No sé cómo hubieran quedado los logos de las cadenas en la aplicación o cómo hubiera empezado a trabajar sin las magnificas EPGs. Gracias.

Para terminar no me olvido de ti. Muchas gracias por esos primeros meses en los que busqué el apoyo y tenías las palabras adecuadas. En los que, pese a todo lo que estaba pasando, me dijiste que no me rindiera, que intentara apartar lo malo y centrarme en los exámenes, que para lo demás ya habría tiempo. Gracias también a tus padres, que siempre me han apoyado. No voy a olvidarte nunca, siempre estás conmigo, si algo he conseguido es tuyo hermano. Gracias Álvaro.

Resumen

La aplicación que se desarrolla en este proyecto es un recomendador de contenidos televisivos. Está desarrollada en Android 4.0 por lo que se puede ejecutar en cualquier smartphone con Android 4.0 o superior y en cualquier tablet con las mismas condiciones de sistema operativo.

Funciona como complemento a Google TV. Una vez que tenemos un dispositivo de Google TV conectado a un descodificador de TDT (Televisión Digital Terrestre) y a nuestro televisor, la aplicación pasa a ser el mando a distancia del dispositivo y nos mostrará la información de la programación a tiempo real. Al mismo tiempo aprenderá qué nos gusta más y qué tipo de programas solemos ver.

Para poder desarrollar una aplicación de segunda pantalla es necesario implementar el protocolo *Anymote-Protocol* mediante la librería*, Anymote-Library* que está a disposición de cualquier desarrollador y lo que nos permite conectar con nuestro dispositivo Google Tv siempre y cuando estemos dentro de la misma red Wi-Fi. Al ser una librería de código abierto puede sufrir los cambios necesarios para adecuar sus características a la aplicación que se está desarrollando.

La información relativa a los programas de televisión se lee de un archivo de EPG. Mientas el usuario la usa, la aplicación aprende de lo que éste ve y concluye qué le gusta y qué no. Así, posteriormente, hace búsquedas en segundo plano gracias a los datos que ya ha aprendido para encontrar contenidos que se emiten de manera periódica y son del interés del usuario, o de contenidos con características en común, como la categoría televisiva a la que pertenecen. El usuario puede configurar si desea recibir una notificación de aquellos programas que le gustan o incluso de los programas similares a los que ha visto.

Google TV es un complemento perfecto para cualquier televisor ya que lo convierte al instante en smart TV. Esta aplicación hace que se pueda aprovechar mejor el tiempo que se dedica a su uso ya que está desarrollada para que sea única por cada usuario, porque cada telespectador tiene unos gustos diferentes.

Palabras clave

Android, smartphone, Tablet, Google Tv, Smart TV, Anymote-Protocol, eclipse, EPG, televisión, TDT, XMLTV, recomendador.

Abstract

The application, which is elaborated in this project, is a recommender of television's broadcast contents. It is developed on Android 4.0 operative system. Consequently, any device, which implements this system, such as smartphone or tablet, would be able to run it.

This application works as Google TV's complement. Once users have a Google TV device connected to a DVB-T (Digital Video Broadcasting Terrestrial) decoder, and also to a TV, the application becomes the remote control and shows us the entire TV programming information on real time. At the same time, it stores what kind of programs users enjoy the most and which ones they usually watch.

Within a Google TV environment, in order to develop a second-screen application, it is necessary to implement the Anymote-Protocol by using the Anymote Library, which is available to anyone, and what allows the user to be directly connected with our own Google TV device, as long as the user stays using the same Wi-Fi network. The advantage of the Anymote Library is its freeware capacity, so users are able to make any change in relation to their own interests.

All the information about TV programming is decoded from an EPG file. While making use of the application, the variety of contents the user is watching is stored, so as to recognize what kind of programs the user like. That is, in a background process, using the information stored, the application finds weekly or daily contents that have been previously watched by the user, or the ones similar in content or category, in order to satisfy their interests. Users can choose if they want to get a notification of the different contents they like the most, or even of a variety of similar programs in comparison with the ones they have already watched.

Google TV is a perfect complement to any TV, since it integration turns a common TV into a smart one. Moreover, it avoids wasting time while watching TV in order to get to something entertained, and offers a wide possibility to customize everything, and makes it unique according to our preferences, since each user has diverse interests.

Keywords

Android, Smartphone, Tablet, Google Tv, Smart TV, Anymote-Protocol, eclipse, EPG, television, DVB-T, XMLTV, recommender.

Índice general

Índice de figuras

Índice de tablas

[Hoja en blanco]

Capítulo 1

1. Introducción y objetivos

Este capítulo sitúa e introduce al lector en el trabajo fin de grado comentando la motivación que lo ha requerido así como los objetivos que lo han impulsado. Para facilitar la lectura del documento se facilita una pequeña descripción de la estructura del mismo.

1.1. Marco y motivación

El consumo de contenidos a través de la televisión no deja de crecer. En España la oferta de TDT (Televisión Digital Terrestre) a día de hoy y a nivel nacional tiene un total de 35 canales de televisión [1] a los que hay que sumar los de ámbito autonómico o territorial, cosa que, a su vez, elevaría la cifra. Cuantitativamente, la oferta de contenidos disponibles en los canales nacionales es enorme, hasta tal punto que todos los contenidos que se emiten de manera simultánea pueden llegar a abrumar al usuario.

Gracias al paso de la televisión analógica a la televisión digital surgió la posibilidad de incluir información adicional en las señales que se emiten para poder dotar al usuario de más posibilidades incluso de interactividad con los contenidos emitidos. Esta última opción no ha llegado a cuajar en el mercado televisivo nacional pero ha continuado evolucionando gracias a Internet.

Internet es por ello, sin duda alguna, un campo en el que la televisión ha de apoyarse para lograr una mejor experiencia de tal modo que no sea un rival sino un aliado. Los televisores de última generación disponen de conectividad a Internet, lo que hace posible que el telespectador tenga a su alcance la información de la red mientras ve la televisión. Además, en los últimos años, el uso de Internet en los smartphones ha crecido de manera exponencial. Desde 2011 su manejo ha aumentado en 20 puntos y el 98% de los usuarios de un teléfono inteligente accede a Internet con él. [2], lo que ha facilitado el acceso a la red para millones de personas, beneficiándose la televisión para lograr un nuevo enfoque. La posibilidad de llevar la televisión a estos dispositivos origina a su vez que los contenidos lleguen más lejos. De este modo, también desde estos dispositivos el usuario puede aprovechar las ventajas de Internet mientras consume televisión.

A la vista de los acontecimientos podemos augurar que el futuro de la televisión va ligado estrechamente a Internet, y que el abanico de posibilidades que con ello se despliega es enorme. Aprovechando así el grado de penetración de los teléfonos inteligentes y el nexo de unión entre la televisión e Internet se desarrollan diferentes aplicaciones para la vida cotidiana que hacen al usuario capaz de aprovechar mejor el tiempo que dedica a ver la televisión. Aplicaciones, muchas de ellas directamente ligadas con los televisores de última generación algo a lo que no todo el mundo tiene acceso.

Disponer de una herramienta como Google TV en un televisor hace que éste se convierta al instante en un televisor de última generación con acceso a

Internet sin realizar una gran inversión. Inversión que se tiene muy en cuenta, pues la compra de un televisor se acomete a largo plazo. Por tanto un usuario de Google TV debería poder beneficiarse de las aplicaciones potenciales para su smartphone que normalmente van ligadas a un determinado televisor.

Aprovechar las múltiples posibilidades que tiene Google TV motiva la acometida del desarrollo de una aplicación para teléfonos con sistema operativo Android que permitan al usuario aprovechar mejor el consumo de televisión. De esta manera, se logra poder dotar a cualquier usuario con un dispositivo Android con la posibilidad de obtener la funcionalidad de nuevas televisiones Smart TV que aprenden de lo que vemos, sin necesidad de invertir en un nuevo televisor, simplemente haciendo un gasto mucho menor en un dispositivo Google TV. Así pues, un usuario podrá hacer un uso del tiempo que dedica a ver la televisión mucho más productivo ya que podrá conocer contenidos que no ha visto anteriormente y que tienen una alta probabilidad de gustarle y además no se olvidará de los programas que se repiten de manera periódica porque la aplicación se encargará de recordar que lo emiten.

A nivel personal desarrollar una aplicación para un sistema móvil supone un auténtico reto debido a las características especiales de los terminales, pese a que día a día van siendo más potentes y eficientes tienen distintas peculiaridades que hace que desarrollar una aplicación que se vaya a ejecutar en ellos requiera una gran dedicación. Lo que supone desarrollar una aplicación en Android es poder mejorar mis conocimientos sobre la plataforma y conocer bien un sistema operativo que marca tendencia en el mercado a día de hoy.

1.2. Objetivos

El objetivo principal del proyecto es diseñar e implementar una aplicación que sirva como segunda pantalla para dispositivos Google TV y permita al usuario manejar su smartphone como mando a distancia para que la aplicación, al mismo tiempo, aprenda sobre el usuario y lo que suele ver.
Para poder llevar a cabo el objetivo principal, el trabajo se ha de fragmentar en pequeños puntos intermedios que han generado los siguientes fines secundarios:

- Profundizar en el sistema operativo Android y todas sus características.
- Conocer la variante de Android desarrollada para dispositivos Google Tv.
    - Profundizar en la librería Anymote que nos permite la comunicación entre Smartphone y Google TV.
- Aprender el tratamiento de archivos XML y su uso y estandarización en EPG.
- Comprender cómo funcionan las tareas en segundo plano en el sistema operativo Android.
- Manejar el entorno de desarrollo Eclipse así como el nuevo entorno propio para Android, Android Studio.

- Profundizar en la compatibilidad de pantallas y las diferentes posibilidades de adecuar a cada dispositivo.

1.3. Fases de desarrollo

La ejecución del proyecto se ha llevado a cabo en diferentes fases, cada una relacionada con una parte individual del proyecto. A continuación se dividen de manera cronológica.

1.3.1. Fase 1: Entendiendo el entorno de desarrollo de Google TV

El entorno de Google TV era completamente nuevo para mí y tuve que leer mucho sobre ello. Google tiene habilitada una sección para desarrolladores en Google TV pero se centra, sobre todo, en aplicaciones que se ejecutan dentro del dispositivo. Sobre las aplicaciones de segunda pantalla la información no es tan detallada porque aún no son muy populares y se actualiza lentamente, hecho que condicionó el comienzo del proyecto.

En un primer momento supuse que la clave del desarrollo consistía en poder emular un dispositivo Google TV desde el entorno de desarrollo (Eclipse) y así poder simular la comunicación con el dispositivo real. El *add-on*[1] Google TV necesario para poder lanzar un emulador Google TV desde eclipse, sólo estaba disponible para Linux pero, pese a hacerlo todo de manera correcta, la máquina virtual que crea el AVD (*Android Virtual Device*), no arrancaba. Gracias a la ayuda del tutor, supimos que los parámetros que se crean automáticamente no son válidos y, debido a esto, hay que cambiarlos a mano.

Pasado todo este tiempo, por fin pudimos arrancar el emulador de Google TV en el AVD. Esto no hizo más que darnos otra sorpresa, pues se descubrió que es imposible simular una conexión con un smartphone o con otro emulador en el que se está ejecutando un smartphone. Poco después la documentación de Google [3] confirmó esta sospecha, por lo que se tuvo que adquirir un dispositivo real para poder realizar pruebas.

1.3.2. Fase 2: Comienzo del desarrollo en Android

Una vez que sabía dónde tenía que centrar los intereses, comencé a desarrollar una sencilla aplicación en Android. El objetivo era crear una interfaz de usuario sencilla que se asemejara a un mando a distancia en la pantalla. El objetivo era, sin tener el dispositivo Google TV, poder simular la conectividad con él a través de una primera interfaz que serviría como base para el desarrollo de la aplicación. Para ello es fundamental el protocolo que implementa Anymote Library, cuyo contenido se debe asociar a cualquier aplicación en segunda pantalla que necesite conectarse con un dispositivo Google TV.

1.3.3. Fase 3: Implementar Anymote Protocol

Después del desarrollo de la interfaz de un sencillo mando a distancia y la adquisición de un dispositivo Google TV real, se conectan para probar y conocer mejor el funcionamiento de la comunicación entre ambos dispositivos.

En este momento se descubre que la comunicación es unidireccional y pese a que con el mando a distancia desarrollado conseguimos cambiar el canal no tenemos medio alguno de saber el canal en el que el usuario se encuentra en ese momento.

Para solucionarlo, se decide crear un archivo de configuración que será rellenado por el usuario y en el que indica el dial asociado a cada canal. De esta manera cuando cambiemos el canal, consultando el archivo que el usuario rellena podemos saber el nombre del canal que se está viendo.

1.3.4. Fase 4: Documentación sobre EPG y tratamiento de XML

Para poder realizar el objetivo principal de la aplicación es necesario realizar consultas a la EPG. Los archivos de EPG son archivos XML escritos bajo un estándar XMLTV. Tras la lectura del capítulo 7 del libro Android 4 Anaya Multimedia, aprendo las diferentes formas que existe en Android para leer archivos XML.

Las consultas de la EPG tendrán dos motivos; mostrar la información del programa actual que se está viendo y consultar búsquedas de los contenidos que haya aprendido la aplicación.

1.3.5. Fase 5: Desarrollo de un algoritmo de aprendizaje

Existe la necesitad de contar con una serie de pautas para que la aplicación pueda desempeñar su función principal. Para ello es necesario desarrollar un pequeño algoritmo de inteligencia artificial mediante el cual la aplicación aprenderá de aquello que el usuario ve en televisión.

Se especifica un método de aprendizaje sobre lo que el usuario ve y también se define un método de búsqueda sobre lo que el usuario ha visto y sobre lo que es similar a lo visto por el usuario.

1.3.6. Fase 6: Pruebas de rendimiento

En este punto la aplicación tiene cuerpo y forma y está capacitada para poder llevar a cabo pruebas de rendimiento sobre la misma. Estas pruebas permiten detectar y solucionar errores y también permiten rediseñar determinadas funciones. Preferiblemente se realizan estas pruebas en un escenario real con un dispositivo Google TV, pero no es imprescindible ya que podemos simular el comportamiento del mismo debido a que la aplicación se desarrolla íntegramente en el dispositivo móvil.

1.3.7. Fase 7: Memoria

La redacción de la presente memoria se lleva a cabo cuando la aplicación tiene un alto porcentaje de desarrollo terminado.

1.4. Medios empleados

Si dividimos entre hardware y software los requerimientos que se han necesitado para llevar a cabo del trabajo fin de grado han sido:

- **Hardware**
  - Portátil Macbook Pro modelo 8,1
  - PC
  - Teléfono móvil Xperia Neo V
  - Tablet Asus Nexus 7
  - Descodificador Vizio Google TV
  - Descodificador Hisense Pulse Google TV
  - Descodificador TDT HDMI Sunstech
  - Televisor/monitor HDMI
- **Software**
  - Sistema operativo Apple OSX 10.8.5
  - Sistema operativo Windows 7
  - Sistema operativo Linux Ubuntu 12.04 LTS
  - Sistema operativo Android 4.0
  - Eclipse Helios 3.6.2
  - ADT Plugin 22.0.1
  - Adobe Photshop CS5
  - WebGrab+Plus V.1.1.1
- **Varios**
  - Conexión a Internet
  - Conexión Internet móvil
  - Señal TDT

1.5. Estructura de la memoria

La presente memoria está dividida en 8 capítulos de los cuales se presenta una breve descripción a continuación.

Capítulo 1. Introducción y objetivos.

Este capítulo introduce al lector en la memoria haciendo una pequeña introducción al mundo de la televisión unida a Internet y explicando los objetivos que se han fijado para el desarrollo. Así mismo, se especifican las fases en las que se ha desarrollado el proyecto y los medios necesarios para llevarlo a cabo.

Capítulo 2. Planteamiento del problema.

Para poner en contexto el desarrollo de la aplicación se hace un pequeño análisis de la situación actual de aquellos puntos con los que el proyecto se relaciona.

Capítulo 3. Diseño de la solución técnica.

En este capítulo se pretende orientar al lector en la manera que ha sido concebida la aplicación. Para ello se explica el diseño de la misma y de las funcionalidades más importantes sin entrar en detalles del código que la implementa.

Capítulo 4. Implementación.

El cuarto capítulo de la memoria explica la implementación del diseño explicado en el capítulo anterior centrándose en aquellos apartados originales y diferenciadores. Sin pretender abrumar al lector con líneas de código, se adjuntan imágenes con el código preciso para entender mejor las explicaciones.

Capítulo 5. Pruebas de rendimiento.

Para poder representar mejor cómo se ha probado la aplicación y qué hipótesis han llevado a esas pruebas en este capítulo se contextualizan y se justifican, así como, se explica qué supusieron a la hora de mejorar la aplicación.

Capítulo 6. Líneas futuras.

Como línea de trabajo a seguir se exponen las ideas que se intentarían llevar a cabo en una hipotética continuación del proyecto.

Capítulo 7. Planificación y presupuesto.

En el séptimo capítulo se resumen las tareas llevadas a cabo para desarrollar todo el trabajo fin de grado así como un desglose del presupuesto que conlleva.

Capítulo 8. Conclusiones.

Para terminar la memoria se exponen las conclusiones personales que ha supuesto terminar este trabajo fin de grado.

Capítulo 2

2. Estado del arte

La totalidad del proyecto gira en torno a Google TV, los smartphones y la televisión. Existe un nexo de unión de estos tres elementos que es el sistema operativo Android. Google TV da la opción a los usuarios de adaptarse a las nuevas tendencias que se están imponiendo en los últimos tiempos y dotar a cualquier televisor de las posibilidades que ofrece un Smart TV. A lo largo de este capítulo se pretende orientar al lector sobre el estado actual del mercado en estos aspectos.

2.1. Smart TV

El término Smart TV proviene del inglés y se traduce como *televisión inteligente*. Los televisores inteligentes son algo reciente dentro de la historia de la televisión, si bien es cierto que ahora mismo es algo más común, no llevan muchos años estabilizados en el mercado e incluso sus usuarios aún no se acostumbran a conectarse a Internet [4]. No se debe confundir el término con HbbTV que es un modelo de televisión hibrida entre televisión digital y televisión IP pero que ofrece la posibilidad de televisión conectada a Internet.

Este tipo de televisores ofrecen Internet dentro del propio televisor, integran un acceso a la red que, combinado con la evolución de la misma, da como resultado un cambio total en la experiencia final del usuario. El hecho de disponer de acceso a Internet dentro del televisor no se debe interpretar como si fuera sinónimo de ver la televisión en el ordenador, son cosas distintas. Las Smart TV van un paso más allá y buscan aprovechar el acceso a Internet para mejorar la experiencia del usuario mientras ve la televisión.

Cuando comenzó el paso de la televisión analógica a la televisión digital en España, en el año 2005,[5] (sin contar la plataforma QuieroTV, 1999 [6]) una de las características destacadas de la nueva TDT era la interactividad, la cual planteaba que el consumidor podría aprovechar la comunicación bidireccional para participar en concursos, dar opiniones o incluso hacer la compra. Esta funcionalidad no llegó a tener éxito entre los usuarios por lo que no se fomentó su desarrollo. Pese a ello, podríamos interpretarlo como un primer paso hacia televisiones que nos permiten ir más allá y usarlas para algo más que ver contenidos.

Personalmente considero que hay dos hechos fundamentales en Internet para que las televisiones se vean obligadas a evolucionar hacia el mundo web.

- Consumo multimedia. En la última década el uso de Internet para consumir contenidos ha aumentado de manera exponencial gracias a plataformas como YouTube, el consumo de vídeo y música a través de la red es algo que se ha convertido en parte del día a día y no parará de crecer a corto plazo [7].

- Las redes sociales. Siendo un pilar de la llamaba web 2.0 [8] el uso de las redes sociales ha adquirido una importancia muy grande dentro del mundo interactivo. Según el IV estudio anual redes sociales del IAB (Interactive Advertising Bureau) [9], 8 de cada 10 internautas de entre 18 y 50 años utilizan las redes sociales en España, lo que supone un gran número de usuarios que generan, a su vez, potenciales usuarios de contenido multimedia debido a que uno de los usos que suele darse a estas redes es la de compartir contenidos de este tipo.

Si consideramos que los primeros orígenes de lo que hoy conocemos como Smart TV fue aquella implementación de una TDT interactiva, el concepto ha evolucionado mucho. Hoy día la televisión inteligente quiere tener esa interactividad y bidireccionalidad con el usuario pero también quiere, e incluso necesita, acoger las dos vertientes más utilizadas en Internet y con ello, y para así poder dar cabida a Internet dentro del televisor, hay que superar una barrera importante; la interfaz.

Acceder a la web desde el ordenador o desde dispositivos móviles es algo a lo que estamos acostumbrados. Manejamos un hardware de comunicación con la máquina que bien puede ser el ratón y el teclado para usar nuestro ordenador o bien nuestros dedos para manejar las pantallas de los dispositivos táctiles. Sin embargo, el hardware clásico por excelencia para manejar la televisión es el mando a distancia, hay que mantener el mando a distancia porque no es factible usar un ratón convencional o un teclado debido a la ubicación histórica de los televisores en el hogar. Se intenta por ello integrar la funcionalidad de un ratón en el mando a distancia mediante *trackpads* similares los de los portátiles o teclados *qwerty* completos en la parte posterior para facilitar la escritura.

Igualmente, se ha experimentado con mandos que incorporan sensores de movimiento para poder navegar por los menús. La interfaz gráfica que aporta el sistema operativo del Smart TV debe ser sencillo, común y factible de manejar con el mando a distancia sin necesidad de incorporarle muchas funcionalidades nuevas. Es pues la barrera del uso algo muy importante a la hora de manejar una Smart TV.

Figura 1. Dorso y anverso del mando del descodificador Hisense Google TV

Los últimos modelos del mercado llegan a ofrecer además aplicaciones personalizadas que recomiendan al usuario contenidos televisivos en función a otros contenidos que ya ha visto con anterioridad. También ofrecen la posibilidad de conectar con las redes sociales y compartir lo que estás viendo de manera rápida y sencilla.

Una Smart TV se convertirá, poco a poco, en algo común en el mercado y de la misma manera los usuarios irán haciéndose con una ya que la tendencia es unificar los productos en uno.

2.2. Google TV

En mayo de 2010 Google presentó en sociedad la plataforma Google TV desarrollado conjuntamente con Sony, Intel y Logitech. [10]

La plataforma está implementada bajo el sistema operativo Android y se basa en la unión de Internet y televisión, pero a diferencia de las Smart TV, da la posibilidad de acceso en cualquier televisor, es decir, da la oportunidad de convertir un televisor normal en un televisor inteligente con las ventajas que ello conlleva.

Google TV fue diseñado para funcionar con un proveedor de contenidos propio o externo, por lo que no obliga a pagar una suscripción mensual después de haber adquirido el dispositivo. Actualmente Google TV sólo está disponible en Estados Unidos, pero es compatible con cualquier sistema de televisión digital, incluso con sistemas de DVD o Blue-Ray vídeo. El esquema de conexión sería el siguiente.

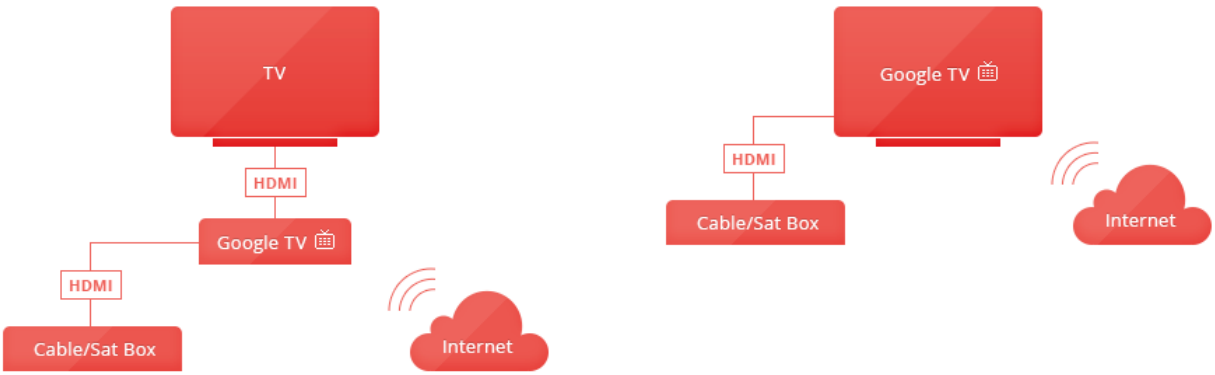

Figura 2. Posibilidades de conexión de Google TV

Los usuarios pueden acceder a los canales lineales de televisión y a la totalidad de contenidos de la web, gracias a que Google TV se implementa sobre el sistema operativo Android e incorpora Google Chrome como navegador. Gracias a las aplicaciones nativas el usuario puede disfrutar de vídeo bajo demanda en streaming con plataformas como Netflix (EE.UU.), Amazon VoD y YouTube, además de tener la posibilidad de utilizar las aplicaciones de Google Play [11].

Las aplicaciones que se ejecutan en Google TV son y pueden ser muy variadas. Al igual que ocurre en los dispositivos móviles las aplicaciones más populares son los juegos así como aquellas aplicaciones que incorporan redes

sociales. No obstante, a la hora de desarrollar una aplicación para Google TV, hay que tener en cuenta que no se trata de un teléfono o una tableta. La filosofía que se sigue con los dispositivos táctiles cambia. Las aplicaciones deben estar diseñadas teniendo en cuenta que serán visualizadas en una televisión, cuyo tamaño mínimo medio estará en torno a las 20 pulgadas por lo que la navegación por menús y la distribución debe ser acorde a ello. A la hora del manejo, el mando a distancia será la opción más factible, lo que limita la navegación por la aplicación a una cruceta de cuatro direcciones (izquierda, derecha, arriba y abajo) o a un trackpad que permite mover un cursor por la pantalla, pero cuya sensibilidad es algo crítico a tener en cuenta. Además, las televisiones suelen situarse a varios metros del usuario, por tanto hay que olvidar cualquier control táctil porque no tiene sentido [12].

Las aplicaciones desarrolladas para Google TV tienen un segundo marco, el navegador web Chrome. Como se ha comentado, gracias a esta plataforma el usuario puede navegar por la web de manera similar a la que navega en un ordenador, pero las páginas webs suelen estar orientadas a pantallas mucho más pequeñas que los televisores y por tanto su manejo y su consumo desde distancias largas puede resultar complicado. Debido a este hecho hay una rama del desarrollo de aplicaciones que se centra en optimizar los sitios webs cuando son mostrados en el navegador Chrome de Google TV. Lo más frecuente es encontrar aplicaciones que presentan páginas web adaptadas y optimizadas a televisores [13].

Como se ha comentado, las aplicaciones deben estar pensadas para que sean manejadas por una interfaz hardware como el mando a distancia, pero siendo un sistema desarrollado en Android, es casi una obligación aprovechar este hecho para generar una comunicación con los dispositivos móviles que utilizan dicho sistema operativo. Este grupo de aplicaciones se agrupa dentro de las llamadas aplicaciones de segunda pantalla (second-screen apps) [14].

2.2.1. Anymote Protocol

Todas las aplicaciones de segunda pantalla tienen algo en común, a parte de estar desarrolladas para tener control sobre algún punto específico de Google TV, para escribirlas deben implementar Anymote Protocol [15].

Este protocolo permite a las aplicaciones establecer una conexión del estilo cliente-servidor, entre el terminal y Google TV. El protocolo implementa un proceso de búsqueda que examina la red local en busca de un dispositivo Google TV, cuando lo encuentra genera un certificado específico para esa aplicación que permite establecer una comunicación entre ellos que no se romperá hasta que se lance una petición expresa para hacerlo. De esta manera la comunicación queda enlazada para el uso de la aplicación. La información intercambiada entre cliente y servidor se protegen usando TSL/SSL para evitar que pueda ser escuchada por terceras aplicaciones. Como se detalla más adelante.

2.2.2. Comunicación

Los mensajes entre la aplicación y el servidor Google TV se implementan utilizando MessageLite Protocol [16] para aprovechar al máximo los recursos de Android y no saturar el sistema ya que es un protocolo muy ligero.
Los mensajes remotos (*Remote Messages*) especifican si es necesario una petición desde el cliente (*RequestMessage*) o una respuesta desde el servidor Google TV (*ResponseMessage*). Si por algún motivo la aplicación intenta enviar un mensaje de respuesta al servidor o el servidor una petición a la aplicación, el mensaje se ignorará. Ejemplos [17].

- **Peticiones**: las peticiones deben tener una estructura similar a la del siguiente ejemplo.

```
message RequestMessage {
  // Message for a key event
  optional KeyEvent key_event_message = 1;
  // Message for a mouse movement
  optional MouseEvent mouse_event_message = 2;
  // Message for a mouse wheel event
  optional MouseWheel mouse_wheel_message = 3;
  // Message containing data
  optional Data data_message = 4;
  // Message send upon connection
  optional Connect connect_message = 5;
  // Fling message
  optional Fling fling_message = 6;
}
```

- **Respuestas**: Google TV mandará respuestas a la aplicación encapsuladas con la siguiente estructura.

```
message ResponseMessage {
// Data message
optional Data data_message = 1;

// Data list message -- ignored
optional DataList data_list_message = 2;

// Fling result
optional FlingResult fling_result_message = 3;
}
```

Las peticiones o las respuestas se generan por la sucesión de diferentes eventos. Estos eventos son generados para crear la interacción entre la aplicación que se ejecuta en el terminal móvil y la aplicación que se ejecuta en el servidor Google TV, o simplemente con el Google TV. Ejemplos de diferentes eventos que pueden darse son los siguientes.

- **Conectar con el dispositivo**: suele generarse para poder enlazar nuestro terminal con Google TV. En este paso nuestro terminal genera una petición en la que se puede describir el dispositivo y el servidor puede mandar una respuesta indicando, por ejemplo, que habilita la conexión.
- **Eventos de teclado o ratón:** nuestra aplicación puede implementar un teclado alfanumérico o un pequeño trackpad. Cuando se accione una de las teclas podrá enviarse un evento diferente asociado a cada una y cuando se desplace el dedo sobre el trackpad se enviará una posición

distinta cada vez. Estos dos eventos son los que más se manejan porque serán los que den la posibilidad de controlar el dispositivo desde el terminal.
En el siguiente ejemplo podemos ver el código asociado a un evento de teclado.

```
// Sends a key event to the server
message KeyEvent {
  // Key code
  required Code keycode = 1;

  // Action (Up/Down)
  required Action action = 2;
}
```

Mientras la comunicación entre cliente y servidor esté activa se puede monitorizar la conexión enviando *ping* desde el cliente. Si la conexión sigue activa el servidor mandará un mensaje de respuesta

2.2.3. TLS/SSL

Como se ha comentado anteriormente, los mensajes que se intercambian entre el servidor y el cliente van protegidos en la capa de transporte por el protocolo de seguridad capa de sockets seguros, SSL (*Secure Socket Layer)* que proporciona comunicación privada entre cliente y servidor dotando de identificación a este último.

El protocolo de seguridad SSL es muy usado en Internet para proteger información, en muchos casos la que está relacionada con transacciones económicas. SSL se implementa sobre TCP (*Transmission Control Protocol*). Este protocolo permite encriptar la información intercambiada entre el cliente y el servidor. Tiene muchas posibilidades y dota de mucha seguridad a las conexiones pero para establecer la conexión con Google TV no se utiliza todo el potencial de SSL. [18]

La seguridad de la conexión se establece en el momento del enlace entre Google TV y nuestro terminal. En ese momento se intercambian diferentes mensajes en lo que se denomina fase de acuerdo, en la que el cliente (nuestro terminal móvil) realiza una petición al servidor (Google TV) y tras una verificación de MAC[2] y aceptar el acuerdo mediante distintos algoritmos envía una clave que se debe introducir en el terminal para cerrar la conexión. Los pasos de la fase de acuerdo se muestran en el siguiente esquema. [19]

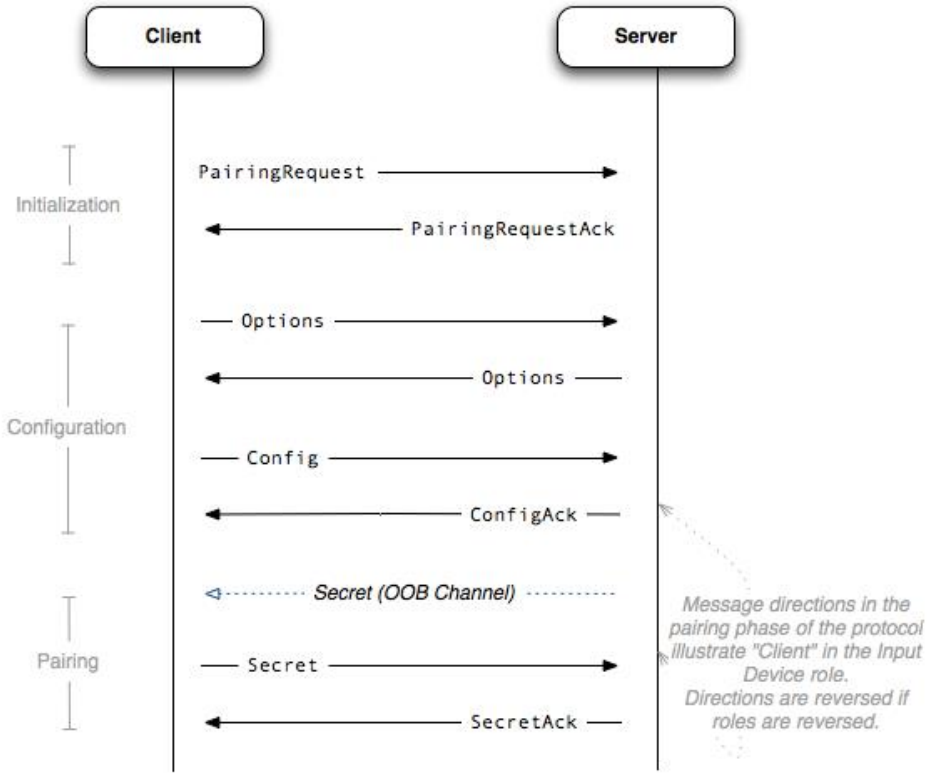

Figura 3. Esquema de intercambio de mensajes para la conexión cliente-servidor

2.3. EPG

La guía electrónica de programa, EPG de sus siglas del inglés *Electronic Program Guide*, es el estándar que el consorcio DVB (*Digital Video Broadcasting*) utiliza para enviar metadatos asociados a cada servicio. Reserva una parte del *stream*[3] de datos encapsulado en un *elementary stream*[4] para enviar la EPG junto al resto de información [20]. Todos los datos son enviados junto a la señal de televisión por lo que el encargado de enviar la EPG es el proveedor de contenidos.

Cuando la televisión aún era analógica ya se enviaba información adicional a la imagen, el teletexto. El teletexto se enviaba en el espacio que no se utilizaba para enviar imagen y quedaba libre. La información era limitada y únicamente informativa, el espectador no podía sacar otra ventaja del teletexto que no fuera información. En televisión digital, el teletexto no desaparece porque es un servicio que no perjudica, pero es relegado a un segundo plano por la guía electrónica de programa, que si bien es una utilidad mejor, no reemplaza al teletexto si no que lo complementa.

Las ventajas que aporta la EPG frente al teletexto se centran sobre todo en la programación. Dentro de la guía electrónica de programa no puede ir otra información que no sea la relativa al contenido que emiten los canales por lo que toda la información que aporta el teletexto en este aspecto no es reemplazable, pero sí que mejora la información de contenidos. Además, la EPG deja de ser una opción de consulta del usuario para convertirse en un elemento más versátil.

La guía electrónica sigue un estándar para almacenar los datos, el XMLTV. Es un estándar que permite estructurar toda la información que aloja la EPG, en ella se puede incorporar títulos, descripciones, información sobre actores, repeticiones… y de manera mucho más accesible que el teletexto. Para que todos los descodificadores de TDT puedan trabajar de la misma manera, el estándar XMLTV facilita el formato XMLTV para que los ficheros XML que alojen información vayan estructurados con el mismo etiquetado y atributos, como vemos en el siguiente ejemplo [21].

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE tv SYSTEM "xmltv.dtd">

<programme start="20080715003000 -0600" stop="20080715010000 -0600"
channel="I10436.labs.zap2it.com">
       <title lang="en">NOW on PBS</title>
       <desc lang="en">Jordan's Queen Rania has made job creation a priority to
help curb the staggering unemployment rates among youths in the Middle
East.</desc>
       <date>20080711</date>
       <category lang="en">Newsmagazine</category>
       <category lang="en">Interview</category>
       <category lang="en">Public affairs</category>
       <category lang="en">Series</category>
       episode-num system="dd_progid">EP01006886.0028</episode-num>
       episode-num system="onscreen">427</episode-num>
       <audio>
              <stereo>stereo</stereo>
       </audio>
       previously-shown start="20080711000000" />
       <subtitles type="teletext" />
<programme>
```

Como advertimos, la información viene encerrada por etiquetas como *programme, title, desc…* que siempre serán iguales. A parte de permitir la lectura de una EPG en cualquier descodificador, da la posibilidad de poder añadir información únicamente añadiendo una etiqueta. Por ejemplo, si es preciso añadir una subcategoría, con utilizar la etiqueta *category* se puede precisar mejor la información.

Una de las mayores consecuencias de esta posibilidad es la ofertas guías electrónicas de programa ampliadas que ofrecen diferentes empresas. En España existen empresas como In&OUT o Cinfo que capturan las EPGs de las señales de televisión y amplían la información, por ejemplo en una película introducen una descripción más precisa o la traducen a otro idioma. Este servicio da como resultado una EPG que ya no tiene la obligación de seguir el estándar XMLTV, por lo que no puede ser enviada por proveedores de contenidos que emitan para descodificadores TDT estándar. Esta EPG mejorada se pone a disposición de empresas privadas de televisión en cuya oferta emiten canales de la TDT española además de otros canales de pago y que para su recepción precisan un descodificador propio, por lo que las características de lectura de los archivos EPG varían y se ajustan a las necesidades de estas empresas.

Se ha comentado que la EPG deja de ser un mero elemento que consulta el usuario, sino que puede tener más funciones. Una es la posibilidad de realizar programaciones. Existen grabadores de vídeo digital también conocidos como PVR (*Personal Video Recorder*) que dan la posibilidad de

realizar grabaciones a futuro usando la EPG. Aprovechando que la información de EPG se envía con una antelación de 15 días, y se va actualizando día a día, también puede utilizarse para buscar contenidos que el usuario quiera filtrar como por ejemplo todos los contenidos dentro de la categoría de series, o incluso hacer esta tarea automáticamente en el descodificador y mostrar al usuario sólo los resultados de aquello similar a lo que ha visto.

En esta última posibilidad se centra el trabajo fin de grado que se describe en esta memoria. Por ello la EPG es un pilar básico del mismo.

2.3.1. Generar un archivo de EPG

Se ha explicado la necesidad de un estándar para escribir la información en los archivos de EPG, el XMLTV, pero aún no se ha indicado nada sobre la forma de generarlos.

Para generar un archivo XMLTV con la información de la programación no hay demasiado desarrollo en el ámbito libre, fuera de las empresas. La señal de TDT se puede decodificar de manera gratuita, y por tanto todo lo que va encerrado en ella también, luego si podemos conseguir extraer la parte de la señal donde se envía la EPG, podemos trabajar con ella. Existen programas, casi todos para un entorno Windows, que realizan esta tarea con mayor o menor éxito. Mientras se investigaba en este aspecto para llevar a cabo la ejecución del trabajo fin de grado, se perfeccionó un programa de libre distribución, WebGrab+ Plus, [22] que realizaba, de manera más que aceptable, la generación de un archivo XMLTV con la información de la EPG.

Es un proyecto que está en constante evolución y no deja de mejorar. Utiliza un algoritmo de búsqueda en diferentes páginas web que alojan la información de las televisiones, como SincroguíaTV para el caso que queramos generar una EPG de la programación española [23]. Dependiendo del uso que se le quiera dar, basta con modificar un archivo de configuración para indicar de qué canales queremos obtener la información.

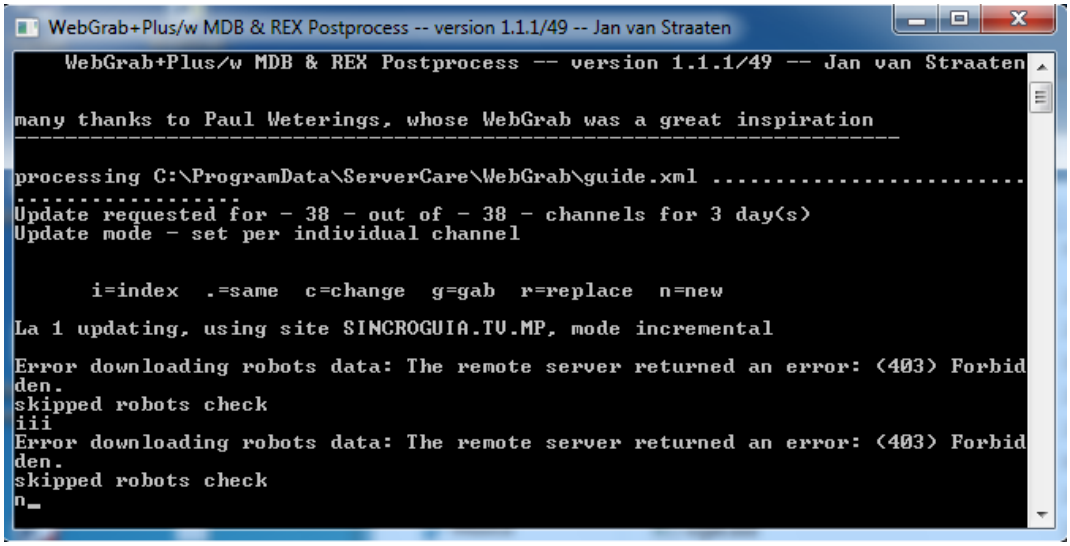

Figura 4. WebGrab+ Plus en ejecución

Para configurar el programa debe editarse un archivo XML indicando los canales que queremos consultar y el portal web. Para cada portal web disponible, WebGrab tiene una librería configurada para acceder a la web en segundo plano. Para este proyecto el archivo de configuración se personaliza con la siguiente lista de canales.

```
<!--Multiplex TVE-->
 <channel update="i" site="sincroguia.tv.mp" site_id="3" xmltv_id="La 1">La 1</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="4" xmltv_id="La 2">La 2</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="505" xmltv_id="Clan TVE">Clan TVE</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="8" xmltv_id="Canal 24 horas">Canal 24 horas</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="9" xmltv_id="Teledeporte">Teledeporte</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="690" xmltv_id="TVE HD">TVE HD</channel>
 <!-- Multiplex Atresmedia -->
 <channel update="i" site="sincroguia.tv.mp" site_id="2" xmltv_id="Antena 3">Antena 3</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="663" xmltv_id="Nitro">Nitro</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="508" xmltv_id="Neox">Neox</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="509" xmltv_id="Nova">Nova</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="510" xmltv_id="laSexta">laSexta</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="673" xmltv_id="laSexta3">laSexta3</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="511" xmltv_id="xplora">xplora</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="670" xmltv_id="Antena 3 HD">Antena 3 HD</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="674" xmltv_id="laSextaHD">laSextaHD</channel>
 <!-- Multiplex Mediaset -->
 <channel update="i" site="sincroguia.tv.mp" site_id="1" xmltv_id="Telecinco">Telecinco</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="507" xmltv_id="La Siete">La Siete</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="665" xmltv_id="Boing">Boing</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="697" xmltv_id="Divinity">Divinity</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="709" xmltv_id="Energy">Energy</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="506" xmltv_id="FDF Telecinco">FDF Telecinco</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="496" xmltv_id="Cuatro">Cuatro</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="669" xmltv_id="Telecinco HD">Telecinco HD</channel>
 <!-- Madrid -->
 <channel update="i" site="sincroguia.tv.mp" site_id="39" xmltv_id="La Otra">La Otra</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="38" xmltv_id="Telemadrid">Telemadrid</channel>
 <!-- Resto -->
 <channel update="i" site="sincroguia.tv.mp" site_id="475" xmltv_id="Discovery MAX">Discovery MAX</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="689" xmltv_id="13tv">13tv</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="159" xmltv_id="MTV España">MTV España</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="516" xmltv_id="Intereconomía TV">Intereconomía TV</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="175" xmltv_id="Disney Channel">Disney Channel</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="664" xmltv_id="Marca TV">Marca TV</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="716" xmltv_id="Paramount Channel">Paramount Channel</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="639" xmltv_id="Kiss tv">Kiss tv</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="189" xmltv_id="Intereconomía Business">Intereconomía Business</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="580" xmltv_id="Aprende Inglés TV">Aprende Inglés TV</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="586" xmltv_id="8Madrid TV">8Madrid TV</channel>

 <!-- Canales de pago PPC-->
 <channel update="i" site="sincroguia.tv.mp" site_id="620" xmltv_id="Gol TV">Gol TV</channel>
 <channel update="i" site="sincroguia.tv.mp" site_id="63" xmltv_id="AXN">AXN</channel>
```

Figura 5. Lista de canales para consultar EPG

Como vemos se ha hecho una selección con una lista de canales que será la que esté disponible en la aplicación. El archivo de configuración también da opciones para cambiar el número de días de EPG que queremos consultar, teniendo un máximo de 15, pero lo más apropiado es consultar de manera diaria para poder tener el archivo generado lo más actualizado posible. Para ello también se ha programado un pequeño script que lanza la tarea de consulta una vez al día, de tal manera que la aplicación acceda a un archivo lo más actualizado posible.

El resultado de las consultas lo podemos ver en este fragmento de EPG, de un archivo XMLTV generado.

```
<programme start="20130805150500 +0200" stop="201307805153000 +0200" channel="Neox">
  <title lang="es">Dos hombres y medio</title>
  <sub-title lang="es">Dum Diddy Dum Diddy Doo</sub-title>
  <desc lang="es">S5E99: A Charlie no le van las cuarentonas porque cree que no tiene nada en común con ellas. Sin embargo, Alan parece
  muy feliz con su nueva novia de 39 y Jake no para de hablar con su novia por teléfono.Charlie acepta una cita a ciegas con una
  amiga de Alan, pero no da la talla porque, definitivamente, es un inmaduro. Herido en su orgullo, Charlie está decidido a
  conquistar a la cuarentona.
  Int.: Charlie Sheen, Jon Cryer, Melanie Lynskey, Holland Taylor, Marin Hinkle, Conchata Ferrell, Angus T. Jones.
  Producción: .
  Año: 2007.
  Edad: TP(c)</desc>
  <credits>
    <actor>Charlie Sheen</actor>
    <actor>Jon Cryer</actor>
    <actor>Melanie Lynskey</actor>
    <actor>Holland Taylor</actor>
    <actor>Marin Hinkle</actor>
    <actor>Conchata Ferrell</actor>
    <actor>Angus T. Jones</actor>
    <producer>Warner Bros. Television</producer>
    <producer>Chuck Lorre Productions</producer>
    <producer>The Tannenbaum Company</producer>
  </credits>
  <date>2007</date>
  <category lang="es">Series</category>
  <category lang="es">Comedia</category>
  <episode-num system="ES">S5E99</episode-num>
  <rating system="ES">
    <value>TP</value>
  </rating>
</programme>
```

Figura 6. Fragmento de EPG

Vemos un fragmento de la programación de un determinado canal, en este caso Neox. Podemos observar las diferentes etiquetas que encierran la información. Estas etiquetas han tenido que ser analizadas para saber qué tipo de información encierran y además cuántas variantes de etiquetas puede haber ya que no todos los programas traen el mismo número de etiquetas y no siempre aparece un mínimo. Todo hay que tenerlo en cuenta para diseñar la lectura de estos archivos.

2.4. Sistemas de recomendación

Los sistemas de recomendación son unas herramientas que permiten mejorar la experiencia del usuario como consumidor de productos. En los últimos años se ha convertido en una pieza clave en el mercado ya que la publicidad, pilar de la economía en Internet, se enfoca cada vez de una manera más personalizada al usuario, y debido a que es utilizada en el ámbito de los negocios, la recomendación ha evolucionado mucho en los últimos años. Para conocer lo bueno o malo que es un sistema habría que medir los resultados subjetivamente dependiendo del usuario.

La función de los sistemas de recomendación es emparejar usuarios con productos. El usuario tiene un papel principal a la hora de la recomendación y sus valoraciones condicionarán los resultados de según qué tipo de sistema de recomendación se use. En la actualidad, existen muchas páginas web que implementan estos sistemas. También hay aplicaciones no tan comunes, ya que se desarrollan para entornos más cerrados, que desempeñan la función de recomendar.

2.4.1. Historia

Desde los años noventa la recomendación se ha ido perfeccionando. Podemos considerar como sistema primitivo de recomendación los primeros filtros aplicados al correo electrónico en los que un usuario podía filtrar determinadas direcciones e impedir la recepción de los mensajes que procedieran de ahí. No es una recomendación como hoy día concebimos pero el resultado final es parecido; el usuario recibe información que le puede interesar.

En la universidad de Minnesota se formó el primer grupo de investigación que desarrolló el filtrado colaborativo [24]. El filtrado colaborativo se basa en la información recibida por varias fuentes sobre cualquier cosa. Todos esos enfoques u opiniones se utilizan para filtrar los datos y mostrar al usuario únicamente aquellos resultados que son más populares entre el resto de usuarios. Pongamos como ejemplo a un grupo de usuarios le gustan las series de televisión y a otro usuario fuera de ese grupo sabemos que le gustan también las series de la televisión. Cuando el usuario aislado busque una película los resultados que se mostrarán en primer lugar serán aquellos que gusten al grupo de usuarios que le gustan las series. Esta técnica se basa en que si dos personas comparten opinión o intereses sobre un tema, es probable que en un tema similar también compartan opinión y gustos. [25]

2.4.2. Algoritmos de recomendación

Los filtros colaborativos u otras técnicas de recomendación son ejecutados mediante una serie de algoritmos que han sido muy evolucionados. [24]

Algoritmo basado en vecinos cercanos

Primer tipo de algoritmo en implementarse para filtros colaborativos. Se necesita una medida que indique el parecido de todos los usuarios con respecto al usuario actual para así establecer una relación entre todos los usuarios. Como medida para calcular la influencia de un usuario B en un usuario A, se pueden utilizar diferentes estadísticos, como el coeficiente de correlación de Pearson con el que obtendremos un peso de influencia de cada usuario en el usuario A. Suponiendo que la relación entre las variables (usuarios) sea lineal, los errores serán independientes y la distribución tendrá varianza y media cero. El suponer estas características ocasiona que los resultados se vean condicionados y no siempre, en la realidad, funcionarán así, pero sí en un alto porcentaje de casos. A parte del coeficiente de correlación de Pearson se pueden utilizar otras medidas de correlación basadas en vectores o en la entropía.

$$W_{a,u} = \frac{\sum_{i=1}^{m} (r_{a,i} - \overline{r_a}) * (r_{u,i} - \overline{r_u})}{\sigma_a \sigma_u}$$

Figura 7. Fórmula para el coeficiente de correlación de Pearson

Estimar una recomendación basándose en los pesos de la correlación es más efectivo cuantas más muestras existan para poder calcular la influencia de unos usuarios en otros. Pongamos un sistema en el que hay pocas muestras (pocos vecinos). En estos casos se suele realizar una selección de vecinos para quedarse con aquellos más representativos como por ejemplo los que superen un determinado umbral o, de otra manera, tomar un número determinado de vecinos cada vez y realizar los cálculos siempre con el mismo número de muestras. Como siempre, la solución no será óptima y habrá que ajustar el sistema a una serie de pautas que se ejecuten dependiendo de cada supuesto.
Para generar una recomendación este algoritmo utiliza una fórmula básica, con la información de cada vecino para proporcionarla con la correlación del usuario actual.

$$p_{a,i} = r_a + \frac{\sum_{u=1}^{n} (r_{u,i} - \overline{r_u}) * W_{a,u}}{\sum_{u=1}^{n} W_{a,u}}$$

Figura 8. Fórmula para ajustar correlaciones

Algoritmo basado en elementos

A diferencia del anterior tipo, éste busca cercanía entre elementos. Hay que seleccionar elementos que un usuario ha votado y después se comprueba lo similar que es con el resto de elementos del sistema. Es decir, mide la correlación que existe entre los gustos de un usuario basándose en los resultados que han generado el resto de usuarios del sistema. La ventaja que aporta este sistema respecto a la comparación de usuarios es la variación. La similitud entre dos elementos es menos variable que la similitud entre dos usuarios permitiendo así agilizar el proceso, puesto que no es necesario comparar a todos los usuarios del sistema.
A fin de comparar elementos se utilizan diferentes técnicas al igual que en el algoritmo de vecinos cercanos se basan en la correlación o en vectores.

Las similitudes basadas en vectores se realizan con el coseno del ángulo que forman. Se considera cada elemento un vector y su similitud dependerá del coseno del ángulo que forman dichos vectores dentro de un especio vectorial de m dimensiones.

$$\cos(x1, x2) = \frac{\vec{x1} \cdot \vec{x2}}{\left\|\vec{x2}\right\| \Delta \left\|\vec{x1}\right\|}$$

Figura 9. Fórmula para calcular el ángulo que forman dos vectores

Las similitudes basadas en correlación también utilizan el coeficiente de correlación de Pearson, como en el caso anterior, salvo que esta vez se compara directamente la valoración que un usuario ha dado a un elemento

$$\text{sen}\,(x1,x2) = \frac{\sum_{u \varepsilon U}(R_{u,x1}-\overline{R}_{x1})(R_{u,x2}-\overline{R}_{x2})}{\sqrt{\sum_{u \varepsilon U}(R_{u,x1}-\overline{R}_{x1})^2}\sqrt{\sum_{u \varepsilon U}(R_{u,x2}-\overline{R}_{x2})^2}}$$

Figura 10. Fórmula para calcular la valoración entre dos usuarios

Para calcular una recomendación se analizan los datos obtenidos. Se pueden sumar las similitudes obtenidas para cada elemento y así obtener un valor de recomendación por elemento cuando se compara con los que, en teoría, serían similares.

Estos dos algoritmos son un pequeño ejemplo sencillo de la estadística usada en los sistemas de recomendación. Actualmente, se ha desarrollado un gran número, consiguiendo resultados muy buenos. Aún así, no siempre la estadística proporciona los mejores resultados. Siendo lo idóneo para obtener algoritmos parametrizados, capaces de ser adaptativos, el factor humano en este tipo de procesos tiene todo el poder de decisión, por lo que es complejo encontrar un punto en el que el acierto sea total. A pesar de ello, los algoritmos de recomendación utilizados hoy día tienen una ventaja a la hora de determinar su validez, y es que siempre muestran unos resultados que condicionan al usuario puesto que a su vez le ocultan otros, y por tanto puede ser que un usuario siempre valore una recomendación de manera positiva porque no conoce todas las opciones reales.

2.4.3. Implementación

Los primeros sistemas de recomendación se han implementado en páginas webs que venden productos. En este tipo de entorno las recomendaciones suelen buscar información de otros usuarios con la que cotejar la información del usuario actual. Para generar una recomendación no se usa exclusivamente la información del usuario ya que se intenta socializar al mismo utilizando los datos de otros usuarios anteriores. Sin embargo, hay sistemas de recomendación que se implementan fuera del ámbito comercial de la web donde es más útil la información del usuario actual que la del resto.

Con el fin de poder recomendar un contenido, se ha de disponer de un etiquetado correcto de los contenidos. Es decir, a un vídeo, una canción, un libro, una página web, se deben asociar palabras, etiquetas, que lo identifiquen, con el objetivo de poder aplicar, posteriormente, alguna técnica de filtrado. En los casos más sencillos de recomendación, las técnicas comparan el perfil de un usuario con los datos almacenados y cuando existe una coincidencia dan un resultado. Es evidente que cuantas más características del usuario se posean y más características de la información estén etiquetadas los resultados serán mejores puesto que los filtros serán más precisos.

Así pues, cuanto mejor entrenada esté una aplicación mejores resultados mostrará. Dependiendo del uso y el fin interesa que muestre unos resultados u otros. Utilicemos la página web YouTube como ejemplo. A dicha web se puede acceder como usuario registrado o anónimo y dependiendo de cómo accedas, ésta te recomendará unos vídeos u otros. Si se accede como un usuario anónimo es posible que sólo se muestre los vídeos más populares en el país desde el que estás accediendo o los canales de los usuarios más valorados. Si por el contrario se accede como usuario registrado, directamente cotejará la base de datos en la que están almacenadas las características de los vídeos que has visto anteriormente, como son la categoría, las reproducciones, los votos positivos o negativos… con otra base de datos que contiene los metadatos de los vídeos. Cuantos más datos del usuario posea YouTube más posibilidades existen de acertar a la hora de recomendar un vídeo.

Para funcionar, los sistemas necesitan recopilar información. Las características que recogen los sistemas de recomendación pueden ser explícitas, implícitas o mixtas.

Explícitas

Las aplicaciones pueden mostrar un listado de características que el usuario debe entrar a valorar según sus preferencias y así obtener información del mismo. Este sistema supone un trabajo extra para el usuario que normalmente no aporta muy buenos resultados. Estos sistemas son los usados, por ejemplo, por sistemas de subscripciones a servicios para enviar publicidad.

Implícitos

Los sistemas implícitos, son transparentes al usuario. Estos sistemas deben avisar al usuario de que su información será almacenada bajo la ley de protección de datos al registrarse. En estos casos cuando el usuario introduce una búsqueda en un formulario o pincha en un enlace la aplicación lo registra y va formando una tabla en la que se generan unos parámetros que definan al usuario. Estas aplicaciones suelen presentar el problema del arranque en frío [26], que impide que funcionen hasta que no han sido usadas un número determinado de veces.

Mixtas

Finalmente, las aplicaciones mixtas ofrecen una primera parte explícita para después ampliar los conocimientos del usuario de manera transparente, De esta manera se soluciona el problema del arranque en frío para los sistemas de recomendación. Cuantas más muestras disponga la aplicación para aprender del usuario mejores resultados mostrará.

El esquema muestra cómo se genera una recomendación según un sistema sencillo en el que se tienen en cuenta datos de un usuario actual y de vecinos cercanos. [27]

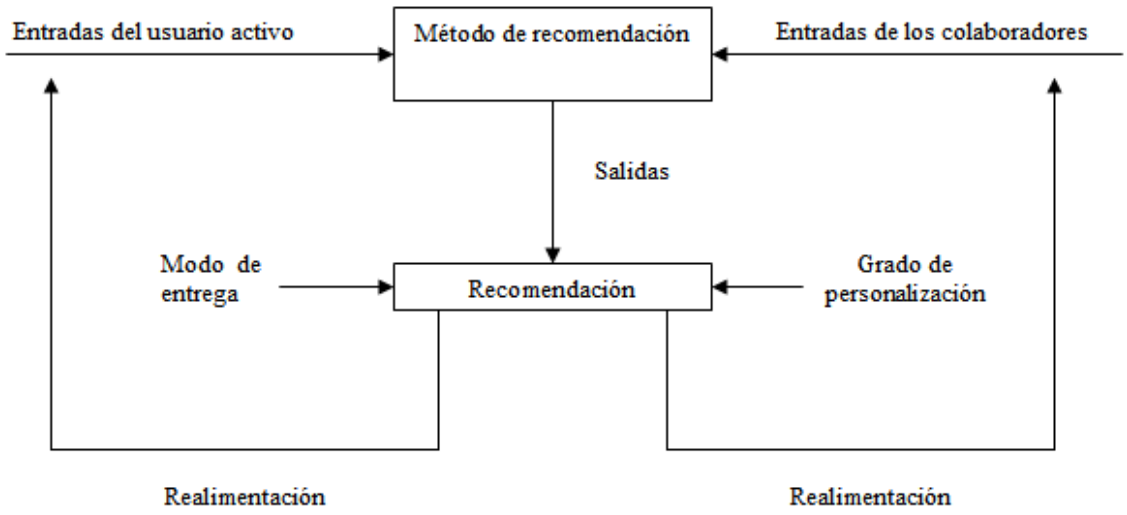

Figura 11. Generación de recomendación.

2.4.4. Ejemplos

A continuación, se introducirán algunos ejemplos de herramientas, páginas web, programas o aplicaciones que utilizan la recomendación para mejorar la experiencia del usuario. Como es lógico, el mecanismo de estos sistemas no es público y es uno de los secretos mejor guardados por cada aplicación, pese a que se puede intuir como funcionan no están abiertos a los usuarios.

YouTube

El mayor portal de vídeos en Internet, es también de los mayores sitios web basado en recomendaciones. Tal y como se ha expuesto anteriormente, YouTube utiliza la recomendación para filtrar los vídeos que muestra a un usuario al entrar en la página o a la hora de estar viendo otro vídeo. Un sistema mixto en el que el usuario tiene su parte activa y una pasiva. También mixto en la comparación de elementos y vecinos ya que utiliza ambas comparaciones.

El usuario no está obligado a indicar si un vídeo le ha gustado o no, o qué categoría le gusta más que otra, pero puede participar de forma activa votando por cada vídeo que ve y simplemente indicar que le ha gustado, sin entrar a valorar más allá ni cuantificarlo.

Como los vídeos están referenciados a unas categorías el sistema de YouTube va aprendiendo los tipos de vídeos que más gustan al usuario. De manera implícita también porque contabiliza el número de visionados que hace a un vídeo o el número de vídeos de la misma categoría que ve a lo largo del mismo día.

Todo este sistema se va haciendo más grande cada vez y se convierte en una herramienta fiable para el marketing y la publicidad. Sabiendo dónde colocar anuncios que se vendan más caros, o sabiendo qué tipo de publicidad introducir en los vídeos que está viendo cada usuario, así como promocionando vídeos recomendados y un largo etcétera.

El sistema de recomendación de YouTube incluye otras muchas variables como la manera de llegar a un vídeo, las búsquedas… y se encuentra en constante evolución.

Amazon

Amazon es una de las mayores páginas web dedicadas a la venta de artículos. Irrumpió en el mercado como una librería digital pero en la actualidad vende multitud de productos. Fue la primera tienda online en utilizar un sistema de recomendación, consiguiendo que, al comprar un producto, al usuario pudiera recomendársele la compra de un producto complementario al que acaba de comprar, estando o no, relacionado.

Su sistema de recomendación se basa en la comparación de elementos y en la comparación de usuarios (vecinos) consiguiendo un resultado muy satisfactorio ya que comparar las compras de los usuarios es algo que suele obtener muy buenos resultados debido, por ejemplo, a que los productos de electrónica normalmente suelen ir acompañados de un producto complementario. Por ejemplo un teléfono suele ir acompañado de una funda para el mismo, y en este sentido el sistema de recomendación de Amazon explota esa característica a la perfección.

La parte implícita del sistema es la más importante. Basándose en las compras que realiza un usuario o en aquellos productos a los que echa un vistazo, el sistema también tiene una parte explícita en la que recoge la información del comprador con los parámetros de búsqueda que ha utilizado. Amazon consigue así una personalización en su página web que es muy importante cuando se trata de una tienda ya que el usuario siempre está viendo productos de los que es un potencial comprador.

Audiovisual

Para la televisión, el cine, la música… Hay muchas aplicaciones y webs que utilizan sistemas de recomendación como IMDB o Lastfm. De la misma manera, hay programas como Spotify que tienen su propio sistema de recomendación. Sin embargo, las aplicaciones para el ámbito televisivo son más escasas, pues suelen desarrollarse en sistemas cerrados.

Las Smart TV más modernas tienen sistemas propios de recomendación al usuario que no se basan en la sociabilidad sino únicamente en lo que el espectador aporta al sistema de manera individual. En este sentido también hay plataformas que implementan estos recomendadores como por ejemplo la plataforma TiVo. En estos *set top boxes* (STBs[5]) se ejecutan aplicaciones que computan lo que el usuario ve para posteriormente recomendar un contenido de la misma categoría o canal.

Estas aplicaciones desarrolladas para entornos cerrados como STB o televisiones no tienen en cuenta, de momento, el factor social. Se han desarrollado pensando exclusivamente en un usuario que va a desarrollar la actividad viendo la televisión mediante plataformas que no todo el mundo lo hace. Es decir, las webs son accesibles para todo el mundo mediante multitud de dispositivos o programas, pero no todo el mundo ve la televisión a través del mismo televisor o plataforma, por tanto son aplicaciones que aún están en una

fase inicial y que necesitan un denominador común para poder ser aplicaciones enteramente sociales.

En este sentido, el proyecto que se ha desarrollado en este trabajo fin de grado pretende sentar la base para una posterior expansión al mundo social. Partiendo de un único usuario que se encuentra viendo contenidos, y desarrollando unas recomendaciones ajustadas al mismo, pero con la ventaja de que al estar desarrollada en un sistema como Android, el entorno es más abierto que si se hubiera de desarrollar la aplicación para un set top box o televisión en concreto.

Capítulo 3

3. Diseño de la solución técnica

En este capítulo se describe cómo ha sido concebida la aplicación desde su planteamiento inicial. No se entra a hablar, en este punto, sobre la implementación o resolución de problemas, pues se hará en el siguiente capítulo.

3.1. Escenario

El proyecto nace de la ocurrencia de crear una aplicación para Google TV. Buscando una idea para desarrollar una aplicación para Google TV se presenta la posibilidad de elaborar la misma, no para el dispositivo de la televisión, sino para smartphones con sistema operativo Android. De esta manera se amplían las posibilidades. Finalmente, se decide desarrollar una aplicación en Android que permita la comunicación con un set top box de Google TV y además funcione como recomendador de contenidos.

El desarrollo de una aplicación de segunda pantalla en un entorno Google TV deberá implementar la aplicación móvil, la comunicación con el set top box, la comunicación con un servidor para acceder a la programación y el tratamiento de la información para desarrollar un algoritmo recomendador. El esquema del escenario en el que se utilizará la aplicación es el siguiente:

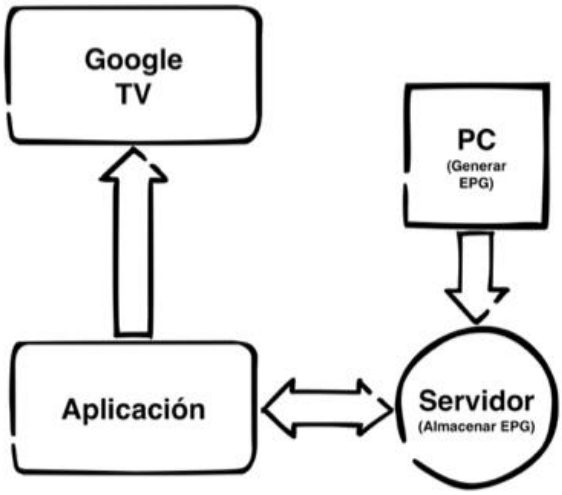

Figura 12. Escenario de uso de la aplicación

La aplicación va dirigida a los usuarios que pretenden optimizar el tiempo que pueden dedicar a ver la televisión de manera que les sea más fácil y directo encontrar un contenido que les vaya a satisfacer, sin tener la sensación de estar perdiendo el tiempo y al mismo tiempo de no olvidar o saber el día en el que emiten el siguiente capítulo de una serie que ya sabe que es de su agrado.

3.2. Requisitos

Para llevar a cabo el proyecto se listan una serie de requisitos que la aplicación debe cumplir, con el fin de desempeñar el objetivo por el que fue diseñada. Este listado permite tener un control sobre las tareas mientras se realiza el proyecto y a su vez permite la creación y el planteamiento de la problemática asociada a cada requisito.

- Comunicación con Google TV.
  - Algo fundamental para la aplicación es manejar bien la comunicación entre el smartphone y el set top box.
- Mando a distancia.
  - La aplicación debe dar la posibilidad de cambiar el canal como si de un mando a distancia se tratara, con el fin de poder usar únicamente el smartphone mientras el usuario ve la televisión y así no estar obligado a usar el mando del propio Google TV.
- Canales.
  - La aplicación debe saber en todo momento en qué canal se encuentra el usuario, de esta manera podrá consultar la programación.
- Programación.
  - Debe crearse un archivo con la información de la EPG para poder consultar en cada momento lo que están emitiendo.
- Consultas al servidor.
  - El archivo no puede alojarse en el móvil debe actualizarse, por tanto se alojará en un servidor al que habrá que realizar consultas vía Internet.
- Algoritmo de aprendizaje.
  - La aplicación debe ser capaz por si sola, de manera implícita, de conocer lo que le gusta a cada usuario para después utilizar esta información.
- Algoritmo de recomendación.
  - Con la información aprendida, la aplicación debe ser capaz de recomendar al usuario aquellos contenidos que puedan gustarle.
- Sociabilidad.
  - La aplicación debe tener un componente social que permita compartir información sobre lo que un usuario ve por televisión.
- Notificaciones.
  - Al realizarse en segundo plano, la recomendación debe ser notificada al usuario de algún modo para que éste se encuentre al corriente de la misma.

Estos requisitos fueron los primeros que se pusieron como base para la aplicación. A lo largo del desarrollo se han visto modificados o ampliados en sí mismos, también se idearon funcionalidades nuevas que no son requisitos mínimos pero que han hecho que la aplicación no se quede en ese punto.

3.3. Estructura

A la hora de llevar a cabo la aplicación y todo su desarrollo en código es conveniente seguir un esquema modular, permitiendo desarrollar diferentes partes de la aplicación sin la necesidad de que todas estén terminadas e ir uniendo las partes como si fuera un puzle. Esta manera de trabajar es la que se sigue cuando se dividen las tareas en pequeños grupos de trabajo. Como este proyecto se ha realizado de manera individual, la ventaja de seguir un diseño modular es poder desarrollar partes diferentes del proyecto simultáneamente. De esta manera, en ocasiones nos encontrábamos con partes que no se podían desarrollar, por ejemplo, porque se dependía de la disponibilidad de un set top box Google TV.

A continuación se presenta un pequeño esquema que indica los módulos en los que se ha dividido la programación del proyecto.

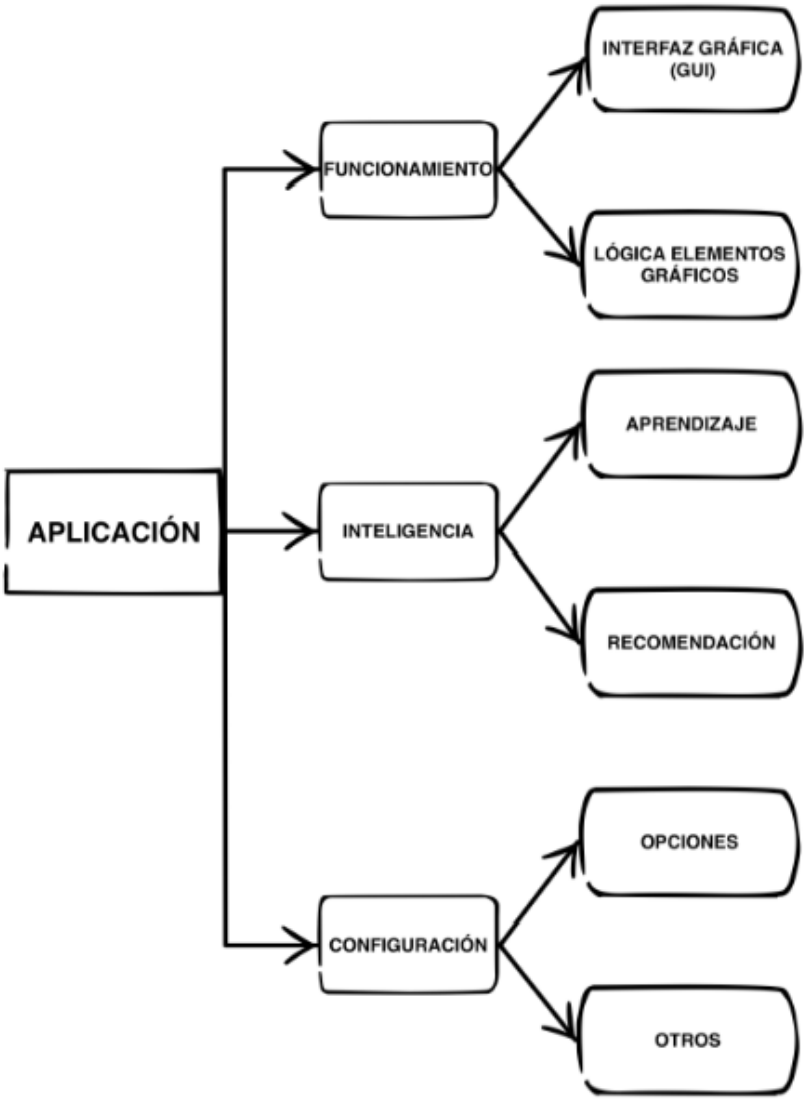

Figura 13. Diagrama de módulos

Como vemos, en este caso, hay tres grandes módulos o bloques que son el de funcionamiento, inteligencia y configuración que engloban las partes más importantes del desarrollo. En el esquema el módulo de funcionamiento y de inteligencia están más juntos entre sí para denotar que están estrechamente relacionados, pues son los pilares de la aplicación. En el módulo de configuración es donde se han desarrollado las tareas más genéricas.

Se ha intentado realizar una división para que cada módulo sea lo más independiente posible, de tal modo, cuando se realicen cambios en un punto de la aplicación, éstos sean lo más transparentes posible para el resto de la misma.

Los módulos en los que se divide una aplicación van asociados, en la mayoría de los casos, a funcionalidad, a los requisitos, es decir, se intenta desarrollar un módulo por cada función o servicio al usuario que deba aportar la aplicación. En nuestro caso, por ejemplo, el módulo de inteligencia es donde se sitúan todas las características de la aplicación relacionadas con el aprendizaje y la recomendación.

3.4. Configuración

Toda buena aplicación tiene que tener un apartado completo de configuración, con el objetivo de adaptarse a las preferencias del usuario. No obstante, en este caso al hablar de configuración en el diseño no se piensa en la personalización sino en el funcionamiento. Para que la aplicación pueda realizar su objetivo principal necesita de una configuración básica.

La aplicación está pensada para que ésta conozca constantemente el canal en el que nos encontramos. La comunicación entre Google TV y el smartphone se asemeja, como ya se ha indicado, a la comunicación cliente-servidor, y si no fuera un entorno cerrado la parte del servidor podría controlarse más. En este caso la parte del servidor es el set top box de Google TV y es un sistema que no puede enviar información al smartphone referida a la señal de televisión que llega al descodificador TDT que está conectado a él. Por tanto Google TV no puede decir al smartphone en qué canal se encuentra el usuario.

Por este motivo es necesario hacer una configuración al instalar la aplicación ideando así un sistema en el que la aplicación sea capaz de saber el canal en el que está el usuario. La solución al problema se solventó con la creación de un archivo de configuración previo al primer uso de la aplicación, en él se debe registrar la lista de canales que tiene le usuario. Se pensó en crear una relación de números con los nombres de cada canal que el usuario se encargaría de rellenar. Dicha solución no es dinámica sino que es estática, por lo que si existe algún cambio en los canales será el usuario el que tenga que modificar ese archivo de configuración. A su vez, si la lista de canales varía en el espectro de la TDT o alguno cambia de nombre es el usuario el que deberá modificarlo o esperar a una revisión. Pese a ser un sistema rudimentario da solución al problema para una primera versión de la aplicación.

Para guardar todos estos datos se utiliza un archivo de preferencias de Android, que al ser una serie de datos variable es lo más apropiado. Una base

de datos no es necesaria porque sólo hace falta asociar dos variables, nombres y números.

3.5. Lógica de aprendizaje

En los sistemas de recomendación, como ya se ha visto y explicado, es necesario un periodo de aprendizaje o entrenamiento durante el cuál, el sistema adquiere conocimientos relacionados con el usuario. En nuestro caso sobre lo que el usuario prefiere ver en televisión.

Para realizar un algoritmo de aprendizaje se pensó desde un primer momento en que debía ser sencillo. De los diferentes métodos existentes para recoger características se partió de la base que la aplicación iba a ser personal y que, asimismo, no debía suponer ningún trabajo para el usuario, decantándose así, por un método implícito. Pese a tener el problema del arranque en frío, un método implícito permite al usuario ser más pragmático y no tener que ingresar diferentes datos.

El algoritmo diseñado es muy sencillo y consiste en lo siguiente. Si el usuario no está viendo la televisión y no cambia de canal en un tiempo determinado, por ejemplo 5 minutos, la aplicación entenderá que lo que se está emitiendo en ese momento atrae la atención del usuario pero no lo interpreta como un gusto, simplemente lo almacena como interés potencial. Cuanto más tiempo pase el usuario viendo diferentes contenidos más información recopilará la aplicación. Si el contenido que está siendo visionado repite alguna característica a algún otro visto con anterioridad se interpretará como interés. Por ejemplo, si el título se repite se interpreta que es un contenido de emisión periódica que el usuario ha visto más de una vez y su categoría y subcategoría puede generar futuros intereses ya que al usuario le pueden gustar contenidos similares.

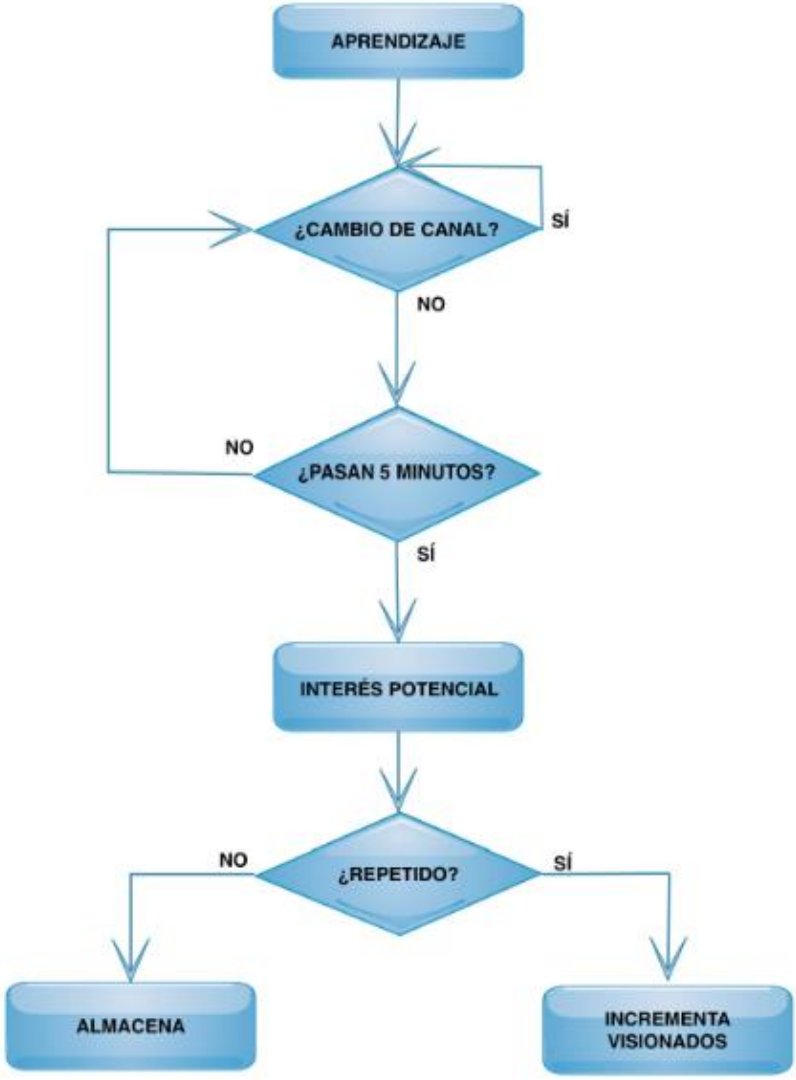

Figura 14. Diagrama de flujo de aprendizaje

Complementario al algoritmo principal se idea una característica de aprendizaje mixto en la que se le da la posibilidad al usuario de indicar a la aplicación que se ha equivocado, es decir, puede haber visto dos veces el mismo contenido pero no necesariamente gustarle o ver que la aplicación le recomienda un contenido y no estar de acuerdo. En este caso la aplicación modificaría los datos almacenados previamente. Si se trata de un nombre en concreto indicaría que no se mostrara como interés, pero si se trata de una recomendación lo que hace el algoritmo es cambiar el peso de las características almacenadas.

Ejemplo. El usuario ha visto una película de acción y la aplicación le muestra otras películas del género que pueden interesarle. El usuario ha indicado que no le gusta un título, la aplicación entenderá que debe darle más peso a otra categoría que no sea acción, pero no eliminarla de la lista por completo.

3.6. Lógica de recomendación

El algoritmo de recomendación es la función principal del proyecto. Tras la etapa de aprendizaje hay que desarrollar un método para poder utilizar los datos recopilados. Para diseñar un algoritmo de recomendación se decidió dividir el trabajo en función de lo aprendido. Según lo que se esté almacenado se realizarán una serie de búsquedas que serán las que lleven a cabo las recomendaciones.

Se diseñan dos tipos de búsquedas: programas repetidos y programas similares.

3.6.1. Programas repetidos

Se define un primer tipo de búsqueda que examinará los datos que ha recopilado la aplicación sobre el usuario y únicamente tendrá en cuenta aquellos que se han visto más de una vez, es decir, entre estos datos principalmente habrá series, magacines, documentales y programas que sean diarios o semanales.

Lo que debe realizar la aplicación es buscar a lo largo de la EPG cuándo vuelven a emitir el programa que ya ha sido visto por el usuario en más de una ocasión y crear una notificación. De esta forma, si el usuario ha olvidado que hoy es el día en el que emiten una serie que le gusta, la aplicación le creará un recordatorio. Del mismo modo, el propio usuario puede deshabilitar esta opción para que no le llegue información que no necesita.

3.6.2. Programas similares

El segundo tipo de búsquedas que se define es lo que realmente se entiende al hablar de sistema de recomendación. Al igual que en el anterior caso se analizan los datos almacenados buscando ya no sólo los contenidos que se han visto en varias ocasiones sino las categorías que están almacenadas.

De todos los datos que la aplicación ha almacenado se entiende que las categorías a las que pertenecen los programas son categorías que pueden suscitar cierto interés en el usuario. Así, por ejemplo, puede que entre los gustos del usuario haya cinco películas y tres sean del género de terror. La aplicación buscará en primer lugar a lo largo de la programación en la EPG si emiten alguna película de ese género, y si encuentra alguna la recomendará al usuario.

La recomendación se basa pues en las categorías asociadas en la EPG a cada programa. Es un pequeño hándicap ya que puede haber contenidos que el usuario puede englobar dentro de una categoría y subcategoría y en la EPG aparecer asociado en otra. Por ejemplo, el usuario engloba algo dentro de *series-infantiles* y en la EPG puede aparecer como *magacín-infantil*. En este caso la aplicación recomendaría contenidos que, quizás, no sean de las preferencias del usuario.

Para dar una futura solución a este problema, hablaremos de ello en el capítulo 6 (Líneas futuras), en el cual se piensa en un posible filtrado por canales que deben estar separados por categorías.

El sistema de recomendación de la aplicación se ha dividido en dos tipos de búsquedas principales. Cuando estas búsquedas obtienen resultados satisfactorios almacenan los mismos para mostrar al usuario todo lo que le pueda interesar. Se idea presentar estos datos separados por categorías para que el usuario tenga un acceso a las recomendaciones más rápido y más sencillo y pueda seleccionar rápidamente aquellas que le interesan de las que no.

3.7. Diseño de las bases de datos

La clave del funcionamiento de la aplicación es el aprendizaje. Ésta debe aprender automáticamente datos relacionados con lo que el usuario ve, y para ello, debe almacenar toda la información para tratarla posteriormente. Para almacenar los datos se ha escogido un sistema de base de datos basado en SQL; SQLite [28]. Funciona muy bien en la plataforma Android. Dentro de la base de datos se ha definido una tabla que tiene los siguientes campos.

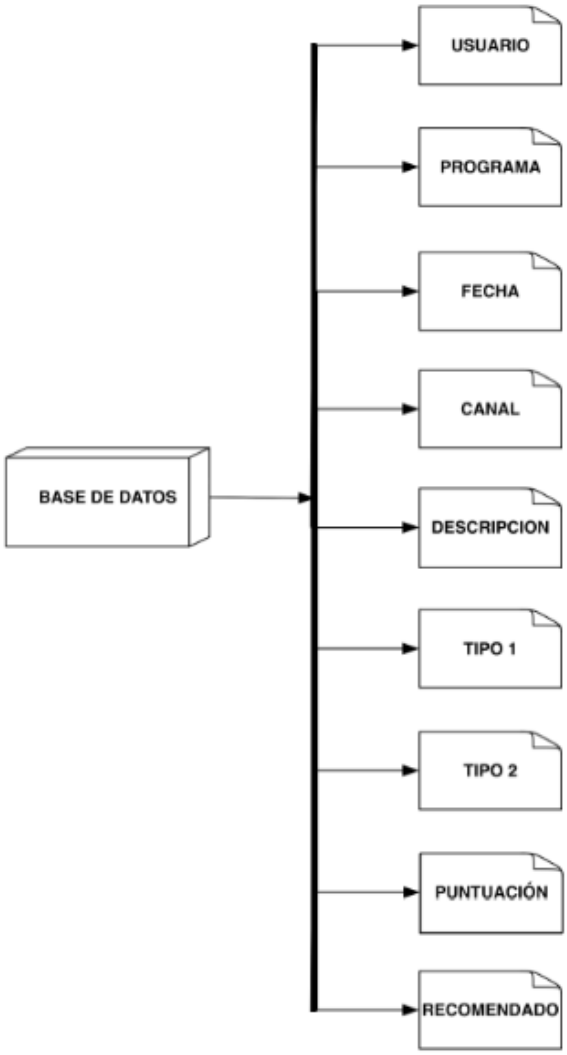

Figura 15. Estructura base de datos de aprendizaje

3.7.1. Campos de la base de datos

Los campos que se han diseñado en la tabla de la base de datos de aprendizaje son los que se muestran en el anterior esquema. Se ha diseñado una base de datos que también pueda servir para futuras mejoras, hay campos que pueden carecer de sentido porque apenas son usados, pero que según vaya mejorando la aplicación se le dará uso y no será necesario rediseñar la estructura de almacenamiento. Para entender un poco mejor por qué se ha diseñado así veamos campo por campo.

- **Usuario:** el nombre del usuario que en ese momento está usando la aplicación, este campo está pensado para explotarlo en futuras versiones añadiendo componentes sociales, por el momento es utilizado para personalizar las notificaciones.
- **Programa:** este campo almacena el nombre del programa que la aplicación entiende que le gusta al usuario. Toma el título según viene en la EPG para que las búsquedas utilicen exactamente los datos tal y como vienen escritos en la EPG.
- **Fecha:** se almacena la fecha en la que se está viendo el programa. En la EPG aparece el día y la hora de inicio y de final, pero no se almacena ese dato, se toma la fecha actual, en el momento que se hace la inserción en la base de datos. De esta manera podemos usar el dato para aprender, también, las horas del día en las que el usuario ve la televisión.
- **Canal:** almacena simplemente el nombre del canal. Es importante ya que las búsquedas pueden verse condicionadas por este parámetro.
- **Descripción:** el campo descripción es el que mayor cantidad de información para el usuario almacena. Únicamente se utiliza para mostrar al usuario la descripción del programa que está viendo, pero también está pensado para usarlo si se diseña un algoritmo que cotejara las descripciones para obtener semejanzas.
- **Tipo 1:** con tipo 1 se refiere a la categoría global del contenido. En las EPGs que se manejan existen 5 grandes tipos de categorías principales, que son: series, cine, magacín, informativos y deportes.
- **Tipo 2:** esta campo almacena las subcategorías. Por cada categoría principal hay una serie de subcategorías que definen mejor el contenido. El número de subcategorías es muy grande por lo que hay que analizar bien la EPG para poder utilizar los datos aquí almacenados.
- **Puntuación:** este campo es un ejemplo de los programados para un futuro uso. La idea es almacenar una puntuación subjetiva del usuario respecto a lo que ve.
- **Recomendado:** si el usuario está viendo algo que se le ha recomendado por la aplicación es necesario marcarlo para obtener aún más información. Por ejemplo para no volver a recomendar lo mismo

3.7.2. Base de datos de recomendaciones y notificaciones

Como ya se ha indicado, los datos que se almacenan en la etapa de aprendizaje son tratados por la etapa de recomendación. Según lo que se haya aprendido se realiza un tipo de búsqueda u otro con el que se pueden obtener resultados. Éstos, son los programas recomendados, generados por los programas aprendidos.

Todos los resultados se almacenan, también, en otra base de datos similar a la de aprendizaje pero en la que se almacenan menos datos, ya que sólo son relevantes los campos de usuario, programa, fecha, canal, descripción, tipo 1 y tipo 2. Salvo en el campo fecha, donde esta vez se almacena la fecha de emisión, el resto de campos almacenan la misma información.

También se diseña una base de datos que almacene los eventos que haya que notificar al usuario, bien porque éste así lo desee o bien porque sea algo automático. Se decide contar con una tercera base de datos y no reutilizar alguna de las ya existentes porque al no tener que crear notificaciones de todo lo que se almacena, se diseña un sistema para crearlas que lea de una base de datos. Así podemos modificar los programas notificados con mayor facilidad.

La aplicación utiliza, así, tres bases de datos de manera simultánea. Una para almacenar los contenidos que gustan al usuario, otra para generar recomendaciones con los datos aprendidos y una tercera para almacenar aquellos programas que se desean notificar. El esquema siguiente muestra la relación existente entre ellas.

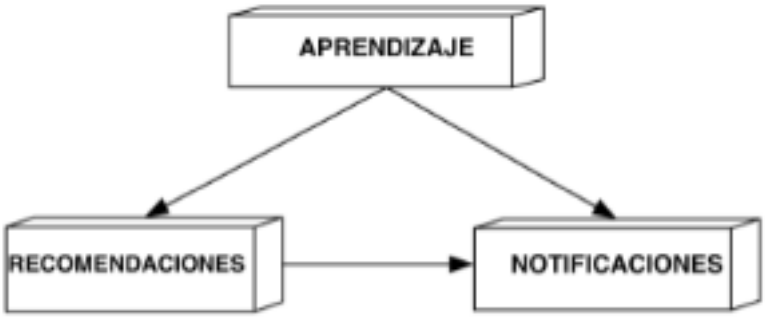

Figura 16. Relación entre las bases de datos

Como se indica en el esquema, las notificaciones pueden ser generadas bien desde los contenidos aprendidos o bien desde los contenidos encontrados en el proceso de recomendación.

Capítulo 4

4. Implementación

A lo largo del cuarto capítulo se explica cómo se ha implementado lo explicado en el capítulo anterior, entrando en los detalles de los problemas que han surgido a la hora de escribir la aplicación y cómo se han resuelto. Si bien no se pretende abrumar al lector con código, se mostrará lo necesario para complementar la información, así como un esquema de las clases java que se han utilizado.

4.1. Interfaz y elementos gráficos

La interfaz gráfica de usuario es el elemento más crítico de cualquier aplicación debido a que es lo que el usuario ve y maneja. Por tanto debe idearse desde una mentalidad sencilla y concreta.

El diseño gráfico de la misma y dotar de funcionalidad a todos los elementos que la conforman fue un gran porcentaje de la aplicación. Desde un primer momento se ideó el diseño de una botonera para asemejar el uso de un mando a distancia. Para llevar a cabo el diseño y la funcionalidad se obtuvo la aplicación Google TV Remote [29] cuya interfaz gráfica es la siguiente.

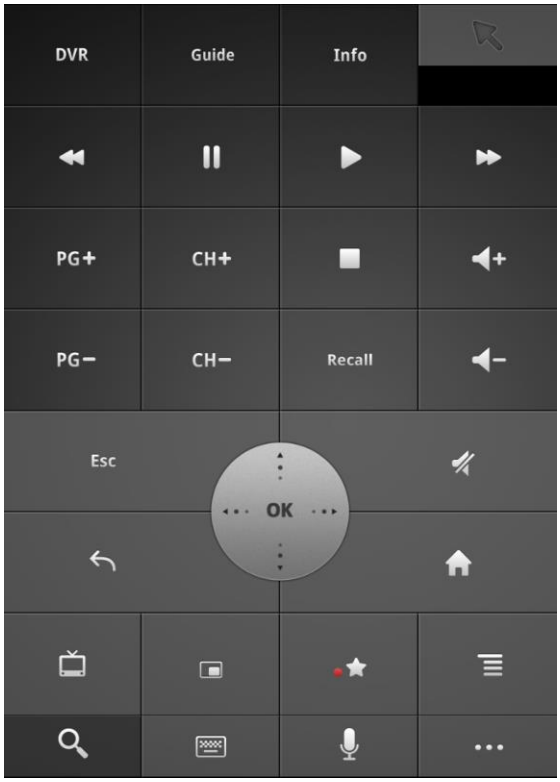

Figura 17. Interfaz gráfica Google TV Remote App

Tal y como se puede apreciar en la imagen, la aplicación presenta una interfaz simple y al mismo tiempo llena de opciones para poder sustituir al

mando de Google TV por nuestro dispositivo Android. Es la aplicación oficial que recomienda instalar Google TV y que ha servido como inspiración.
Si nos fijamos en la aplicación, en ella no aparecen números para cambiar de canal directamente, si no que se presentan las teclas CH+ y CH- para ir de uno en uno. En la interfaz de Qechan, es más lógico que exista una botonera con la que ir directamente a un canal concreto.

Como hemos visto, el código de Google TV Remote es abierto por lo que fue analizado para comprobar cómo realizaba la conexión con el set top box y los diferentes cambios de canal. Para ello no importa la librería Anymote, directamente implementa los métodos y las interfaces dentro de la propia aplicación, algo que incrementa, considerablemente, el tamaño del código. Para utilizar Anymote Library se decidiría importar la librería en Eclipse.

La idea básica de la interfaz, sin pensar en cómo implementarla tenía el siguiente aspecto.

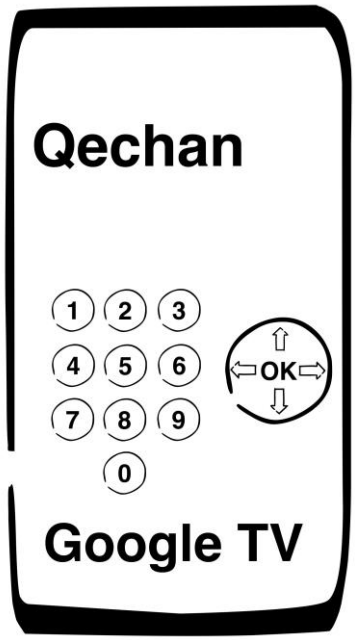

Figura 18. Boceto inicial de la GUI

Al diseñarla se comenzó utilizando el elemento *Button* para definir los botones de los números, pero se descartó la idea y se pasó a usar *ImageButton* ya que permitían mayor versatilidad para incluir una imagen de fondo y así presentar un mejor diseño. Las imágenes que dan vistosidad a los botones están diseñadas con Photoshop.
Gracias a que Android permite diseñar estilos propios para los elementos gráficos según el comportamiento de los mismos, se aprovecha para crear un estilo que cambie la imagen del botón cuando se pulse, de tal modo el usuario sabrá que ha pulsado uno de los botones, porque éstos, cambiarán de color.

Los estilos en Android se definen en un XML plano, que posteriormente se asocia al botón deseado cuando se define en el XML gráfico. A continuación se muestra el estilo definido para el botón correspondiente a la tecla ‘1’, que utilizará el mismo esqueleto para el resto de botones. Al especificar una imagen

particular en cada caso, se necesita un archivo que especifique el estilo para cada botón.

```
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="false" android:drawable="@drawable/b_1" />
    <item android:state_pressed="true" android:drawable="@drawable/p_1" />
    <shape xmlns:android="http://schemas.android.com/apk/res/android" android:shape="rectangle" >
    <corners android:bottomRightRadius="10dp" android:bottomLeftRadius="10dp"
        android:topLeftRadius="10dp" android:topRightRadius="10dp"/>
    </shape>
</selector>
```

Figura 19. Código XML que define el estilo

Si analizamos el XML vemos que se establece una variable que comprueba si el estado del botón es pulsado o no. Según esté, la imagen que dejará ver el botón será una u otra, de ese modo el usuario sabrá si ha pulsado el botón, porque durante un instante dicho botón cambia de color, lo que en realidad ocurre es que se intercambian estas dos imágenes.

Figura 20. Imagen b_1 e imagen p_1

Aprovechando esta característica comprobamos que el número cambia de color. Se diseñó una serie de imágenes para todos los botones cambiaran el color del mismo modo. Como hemos indicado es necesario un archivo de estilos para cada botón. Posteriormente el botón es definido en el *layout*[6] que le corresponda y dentro de su definición, se utiliza el parámetro *android:background* para asociar el archivo de estilo. Así un botón se ha definido de la siguiente manera.

```
<ImageButton
    android:id="@+id/imageButton1"
    android:contentDescription="@string/description"
    android:layout_column="0"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:description="@string/description"
    android:onClick="getChannelChange"
    android:background="@drawable/button_style_1" />
```

Figura 21. Código para definir un ImageButton

El boceto inicial se diseñó para que el mando a distancia siempre estuviera visible, pero se pensó que para aprovechar la pantalla del teléfono y no comprimir los botones, el teclado que hace las veces de mando podría esconderse y aparecer únicamente cuando fuera a usarse. Para satisfacer este diseño se usó un *layout* de tipo tabla en el que cada celda fuese una tecla del mando. Para facilitar el diseño y la visibilidad, los botones pasaron a ser

cuadrados en vez de redondos. Además se decidió que un botón cruceta (D-Pad[7]) no era necesario porque el mando a distancia era únicamente una parte de la aplicación, no tiene que dar servicio al resto de opciones de Google TV por lo que se aprovecha el espacio que deja para añadir botones con más sentido como subir y bajar el canal o el volumen. Para esconder el mando se usó *sliding drawer* [30].

La siguiente funcionalidad que debe cubrir la interfaz gráfica es mostrar la información del programa en emisión. Se pensó en un simple *Textview* con desplazamiento habilitado, así podríamos poner el nombre del programa que el usuario está viendo y una pequeña descripción. Para obtener mayor información sólo cuando se quiera, simplemente bastaría con tocar el textview creado para mostrar la información completa.

Estos dos elementos cubren las funciones básicas de la pantalla principal. Para mostrar al usuario las recomendaciones y aquellos programas que le gustan se pensó en diseñar otras dos pantallas diferentes a las que se puede acceder desde la pantalla principal. De esta forma, queda definido que toda la información adicional se muestre con diálogos o pantallas secundarias con el fin de no abrumar con demasiada información y, del mismo modo, conseguir una interfaz sencilla y práctica. Por ejemplo, la pantalla que muestra las recomendaciones se divide en diferentes pestañas, siendo cada pestaña una categoría donde aparecerán las posibles recomendaciones, según cada categoría, que puedan gustar al usuario.

Figura 22. Aspecto de la aplicación en funcionamiento

4.2. Lectura de archivos de EPG

Android presenta varias opciones para leer archivos XML, ya sean archivos que se encuentran almacenados en el propio dispositivo o bien alojados en algún servidor. Los archivos de EPG, pese a construirse bajo el formato XMLTV, no dejan de ser archivos XML estructurados por etiquetas como se ha podido ver en capítulos anteriores, la peculiaridad es que, pese a estar definidas todas las etiquetas, la lectura de los archivos puede ser algo bastante costoso para la aplicación.

Como ya se ha dicho, en Android existen métodos para analizar (*parsear*) XML, DOM, SAX, XmlPull… Para realizar el análisis de nuestros archivos se comenzó usando la técnica XmlPull [31] ya que permite detener la lectura del XML cuando hemos detectado lo que queremos. Pese a ser algo que resulta lógico al buscar un programa en concreto, no se llevo a cabo debido a que los parámetros de búsqueda no siempre son fijos y puede interesar leer los archivos por completo.

De tal modo, se implementó SAX (*Simple Api for XML*) [32]. Este método analítico requiere un análisis exhaustivo de los ficheros que van a ser leídos para saber lo que se quiere buscar y lo que no. Es decir, requiere un estudio previo por parte del programador donde se generan multitud de archivos de EPG, para comprobar las diferentes variantes que pueden existir y así poder trabajar con las etiquetas. El método SAX es un método intuitivo, mejor que el método DOM (*Document Object Model*) para los archivos que tienen muchas líneas puesto que no es necesaria una lectura previa antes de realizar un análisis, puede ser simultáneo (los archivos de EPG que se manejan en el proyecto pueden superar las 4000 líneas).

Para implementar el método SAX ha sido necesaria la creación de tres clases que funcionen de manera sincronizada. Para comprender mejor cómo funciona, a continuación, se detalla cada una de ellas.

4.2.1. Event.java

La clase Event es la más sencilla. En ella se define lo que quiere leerse del archivo de EPG. Es decir, el resultado de la lectura será una instancia (objeto) de la clase Event, será un evento que tendrá una serie de características, que a su vez está definida en los métodos *set* y *get*[8]. Cada atributo o característica se corresponde con información del evento (programa) que nos resulta relevante como son los siguientes: canal, fecha de inicio, fecha de fin, duración, nombre en castellano, nombre en versión original, descripción, categoría, subcategoría y nombre del capítulo de ser una serie.

Para dar valor a cada atributo se definen los métodos *set*, se utilizan mientras se lee la EPG. Los métodos *get*, como es lógico, se usan cuando quiere conocerse el valor de los atributos, y se usan cuando tratamos la información obtenida.

Si utilizamos el fragmento de EPG de la figura 6 a modo de ejemplo y creamos un objeto de tipo Event en el que se rellenaran sus atributos, obtendríamos lo siguiente.

```
//evento es el objeto de tipo Event
String canal = evento.getChannel() //Neox
String fecha_inicio = evento.getEventBeginTime(); // 20130805190000
```

La ventaja que supone almacenar la información en objetos es clara. No obstante, en este caso particular supone que los datos recogidos puedan ser tratados en otras partes de la aplicación y no necesariamente cuando son leídos. El proceso de lectura es bastante costoso y no conviene introducir tareas extra. A la hora de manejar la información almacenada en un evento, debemos tener cuidado. Normalmente las EPGs presentan problemas en los textos, por ejemplo espacios en blanco, saltos de línea o signos de puntuación donde no existe la necesidad de emplearlos. Al realizarse un estudio de la información que presenta y se observa que en casi la totalidad de los casos estos errores se repiten por lo que se eliminan para que la información mostrada no parezca mal escrita. Este problema es intrínseco de usar un proveedor de EPG gratuito, ya que si la EPG pudiera ser tratada en un paso previo se evitaría tener que editar la información de manera genérica lo que supone ciertos errores. También se puede observar que los archivos que se tratan no presentan un campo de descripción corta, a diferencia del que presentan los archivos que provienen de empresas proveedoras de este servicio. El campo de descripción corta es muy útil para no saturar al usuario con demasiada información en la ventana principal de la aplicación, por lo que se decide mostrar únicamente los 200 primeros caracteres, en caso de tener más, de una descripción. Si el usuario necesita obtener más información acerca del contenido, se le muestra como extra. Esto son sólo unos pequeños ejemplos del tratamiento que se hace a lo extraído de las EPGs, de ahí que tener el acceso en una instancia permita un acceso global a lo largo de todo el programa sin necesidad de almacenar en variables temporales.

4.2.2. EPGParserSax.java

La estructura de esta clase es sencilla. Debe implementar mínimo un método que sirva de unión con la clase donde se define la lectura del XML (EpgHandler.java). Este método es el encargado de devolver una lista de objetos de tipo Event, que contendrán la información deseada.

Cuando se define por primera vez esta clase se utiliza un constructor donde se inicializan los atributos *URL*[9] que iniciamos con la dirección donde se encuentra el archivo de EPG y otro atributo *channel_name* que se asocia con el nombre del canal que estamos viendo actualmente. Lo que es suficiente cuando queremos realizar búsquedas sobre lo que están echando en el canal determinado justo en el instante que se está viendo.
Según avanzaba el proyecto, las necesidades de buscar en los archivos de EPG cambiaron. Se necesita hacer búsquedas complementarias para satisfacer el algoritmo de recomendación diseñado. Estás búsquedas difieren

de la búsqueda de información actual sólo en pequeños aspectos, por lo que no se tiene en cuenta la posibilidad de diseñar nuevas clases para cada tipo de búsqueda, lo que incrementa la complejidad del diseño. Esos pequeños aspectos se ven traducidos en añadir simplemente seis nuevos atributos a la clase ya definida, EPGParserSax.java, atributos que son necesarios para poder filtrar las búsquedas.

Para que sea posible diseñar todas las búsquedas utilizando un solo sistema de tres clases, se aprovecha una característica de Java, la sobrecarga (*overloading*) que permite programar métodos con el mismo nombre pero con una serie de argumentos distintos. Esto hace posible que se definan tres constructores diferentes de la clase, uno para cada tipo de búsqueda, y así según con el número o tipo de argumentos que instanciemos el objeto, se podrá saber el tipo de búsqueda que se desea iniciar.

Según el constructor al que se invoque cuando se crea un objeto de tipo EpgParserSax definimos una variable que tendrá un valor u otro. De esta manera al invocar al método *parse* de la clase, lo primero que se ejecuta es una comprobación de la variable. Ésta puede tener tres valores diferentes: “actual”, “type_1” y “type_2” dependiendo del resultado de la comprobación el método invoca al *handler* de una manera u otra. La invocación se realiza utilizando la librería XML de Android donde se define un método *parse* para XML en cual se ejecuta con tres parámetros: el archivo, la codificación del mismo y el objeto de la clase EPGHandler.

```
Xml.parse(this.getInputStream(),Xml.Encoding.UTF_8, epg_handler);
```

Figura 23. Código para invocar al Handler

El método *getInputStream* es un método necesario para abrir el archivo que se encuentra en la URL facilitada.

4.2.3. EpgHandler.java

Las tres clases definidas para leer un archivo XML mediante SAX son necesarias, pero ésta es donde se define el sentido de la lectura, es decir, se debe implementar qué queremos y qué no queremos leer del archivo.

La clase EpgHandler debe heredar de la clase *DefaultHandler* [33] de Android. El análisis del método SAX se basa en eventos que ocurren según va leyendo el archivo que suelen estar asociados a las diferentes etiquetas que componen el archivo XML que queremos leer. Los eventos pueden indicar el comienzo o final del documento o de un elemento aislado. Al heredar de esta clase, se deben sobrescribir los métodos que implementa la clase *DefaultHandler* para adaptarlos a las etiquetas que contenga el documento.

Como las clases EpgParserSax y EpgHandler están ligadas una a la otra, haber usado sobrecarga para definir los constructores de EpgParserSax impone usar la misma técnica en ésta. Al igual que en el caso anterior se define un constructor por tipo de búsqueda. Se sobrescriben los métodos comienzo y fin de documento y comienzo y fin de elemento.

El método comienzo de documento siempre implementa lo mismo independientemente del tipo de búsqueda que se esté llevando a cabo. Lo que

debe hacer es crear una lista de objetos del tipo Event. En el método fin del documento no implementamos nada, pero se podría hacer, aunque en este proyecto no se debe realizar ninguna acción al terminar de leer el archivo.

A continuación se pasa a describir la implementación requerida para cada tipo de búsqueda en la EPG, especificando en cada caso las diferencias que suponen en los métodos comienzo y fin de elemento, ya que los que controlan el comienzo o fin de documento son comunes para los tres tipos de búsqueda.

Búsqueda de la programación actual

Este tipo de búsqueda se realiza cada vez que el usuario cambia el canal, automáticamente se comienza a buscar en la EPG qué programa corresponde al canal que el usuario está viendo en ese instante. Para ello es indispensable saber dos cosas, el nombre del canal visualizado y la fecha.

Como ya se ha dicho, la lectura del archivo es secuencial leyendo las etiquetas del archivo desde el principio y analizándolo al mismo tiempo. Siguiendo el ejemplo de EPG de la figura 6, observamos que todos los eventos van colocados entre la etiqueta *programme,* por tanto el comienzo de un elemento lo marca esta etiqueta. Cada vez que se detecta, se leen los atributos asociados obteniendo la fecha de inicio y fin del evento y el canal donde se emite. Para filtrar todos los eventos y únicamente mostrar el actual, se utilizan los dos datos que se saben del usuario, el canal que está viendo y la fecha, y con ello se realiza la siguiente comprobación: si el canal que está viendo coincide con el leído se compara la fecha actual y si ésta es mayor o igual que la fecha de inicio y menor o igual que la fecha de final del evento, obtenemos el programa que están emitiendo. Si esto no se cumple, pasamos al siguiente evento.

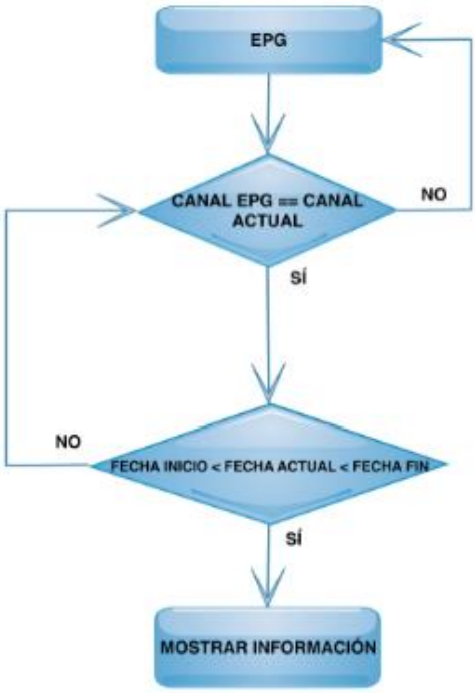

Figura 24. Diagrama de flujo de la búsqueda actual

Cuando se obtiene el programa actual se procede a leer toda la información asociada a él. En el método final de evento es necesario indicar qué hacer con toda la información leída, y según la etiqueta almacenamos la información en el objeto de tipo Event.

```
if (localName.equals("title") && (esp == 0)) {
    currentEvent.setProgramNameEsp(sbTexto.toString().replaceAll("[\n\r]","").trim());
    esp = 1;
}
else if (localName.equals("title") && (esp!=0)) {
    currentEvent.setProgramNameOrg(sbTexto.toString().replaceAll("[\n\r]","").trim());

}
else if(localName.equals("sub-title")){
    currentEvent.setProgramChapterTitle(sbTexto.toString().replaceAll("[\n\r]","").trim());
}
else if (localName.equals("desc")) {
    currentEvent.setProgramDescription(sbTexto.toString().replaceAll("[\n\r]","").trim());

}
else if ((cat==0) && (localName.equals("category"))) {
    currentEvent.setProgramType1(sbTexto.toString().replaceAll("[\n\r]","").trim());
    cat=1;
}
else if ((cat!=0) && (localName.equals("category"))) {
    currentEvent.setProgramType2(sbTexto.toString().replaceAll("[\n\r]","").trim());
}else if (localName.equals("programme")) {
    currentEvent.setChannel(channelName);
    events.add(currentEvent);
}
```

Figura 25. Código para guardar la información del programa actual

Búsqueda de un programa en concreto

Cuando la aplicación ha almacenado programas que se repiten, los que hemos llamado programas periódicos, como un magacín, una serie, un evento deportivo… es posible que el usuario quiera ser informado sobre cuándo se emiten este tipo de programas, porque la aplicación ha aprendido que lo ha visto más de una vez. Para este caso las búsquedas se realizan con una pauta fija: el nombre del evento.

Este tipo de búsqueda no es tan rápida como la búsqueda de la programación actual porque las condiciones de parada no están en la primera línea del archivo EPG sino que se debe entrar en todos los programas hasta llegar al nombre. Esto ocurre porque es el nombre el parámetro principal para detener la búsqueda y así obtener un resultado.

Aunque no sólo el nombre se debe usar, ya que puede ser que obtengamos una falsa alarma y demos por bueno un resultado sin ser el que realmente buscamos. Como es el caso de encontrar coincidencia en el nombre de un evento, pero resulta ser el mismo evento que ha visto el usuario y es el que ha generado el aprendizaje de la aplicación, no uno nuevo. Para solucionarlo deben utilizarse más parámetros de búsqueda como la fecha y el canal.

Teniendo en cuenta todos esos parámetros, este tipo de búsqueda funciona de la siguiente manera.

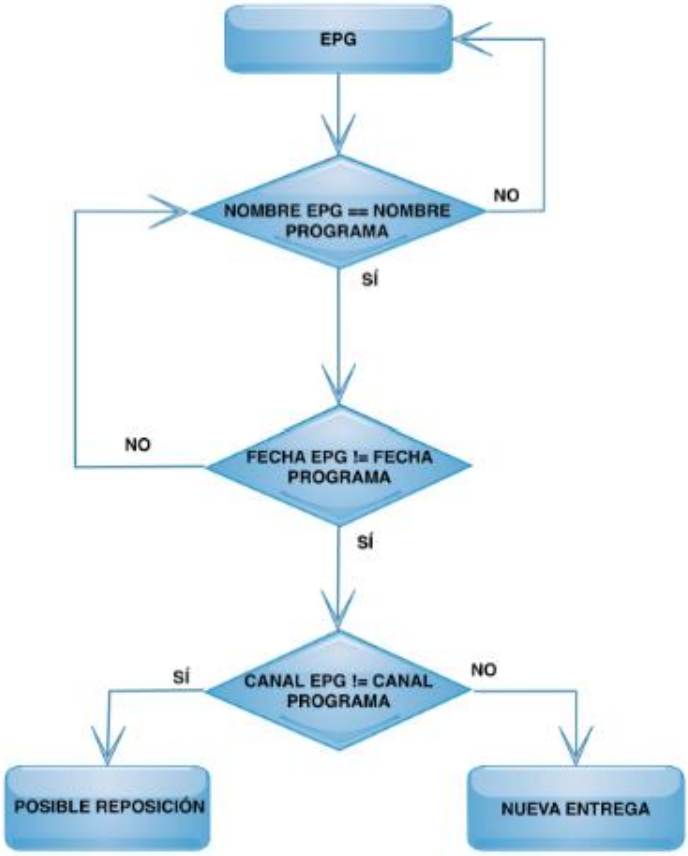

Figura 26. Diagrama de flujo de búsqueda de un programa

Para poder reutilizar la clase EpgHandler se crea una instancia de la clase EpgParserSax con un parámetro que indica el tipo de búsqueda a realizar, utilizando la sobrecarga como se ha explicado anteriormente.

```
EpgParserSax saxparser = new EpgParserSax(programas_periodicos.get(i),
programas_periodicos.get(i+1),programas_periodicos.get(i+2), "type_1");
```

Figura 27. Objeto para buscar un programa en concreto

El objeto saxparser lleva el nombre del programa que se repite, la fecha en la que fue visto, el canal y el argumento “type_1”, que indica el tipo de búsqueda a realizar, en este caso significa que es una búsqueda de un programa periódico concreto. De esta manera en la clase EpgHandler, que es donde hemos definido la búsqueda, se realiza una búsqueda acorde a dichos parámetros. Según se analiza el archivo de EPG si se encuentra un resultado satisfactorio se almacena en un objeto de tipo Event. Los datos que contiene este objeto son tratados al finalizar la búsqueda y se almacenan en una base de datos donde se encuentran todos los contenidos que, posteriormente, se notificarán al usuario.

Búsqueda de una recomendación

El tercer tipo de búsqueda diseñado en la aplicación es la búsqueda de recomendaciones y sigue un esquema similar a la búsqueda de programas concretos, con la salvedad de que, en vez de tomar el patrón de búsqueda basado en un nombre de programa, se basa en las diferentes categorías de los contenidos que la aplicación ha ido almacenando y que supone que al usuario le gustan. Por tanto es de esperar que si le gustan muchos programas de la misma categoría, le guste también alguno que no haya visto de dicha categoría.

Para diseñar la búsqueda primero se revisan las categorías principales. En esta versión inicial se tienen en cuenta las siguientes: series, cine, deporte y magacín. Si hay almacenado algún contenido dentro de esas categorías se procede a examinar la subcategoría asociada al mismo y se crea una instancia de la clase EpgParserSax para poder realizar la búsqueda en la EPG. Por ejemplo, para buscar contenidos relacionados con el cine se realiza de la siguiente forma:

```
EpgParserSax saxparser = new EpgParserSax("Cine",sub_cine.get(k),
        fechaConsulta,"type_2");
```

Figura 28. Objeto para buscar recomendaciones de cine.

Los parámetros necesarios en el tercer tipo de sobrecarga son la categoría, la subcategoría, la fecha en la que se realiza la búsqueda y el tipo de búsqueda. Es necesario saber cuándo se realiza la búsqueda porque al no tener ninguna referencia temporal no se debe recomendar un contenido que ya se haya emitido. Las subcategorías elegidas serán aquellas que más se repitan. Con el objetivo de lograr un mayor porcentaje de éxito, se debe llevar a cabo un análisis de los datos previamente.

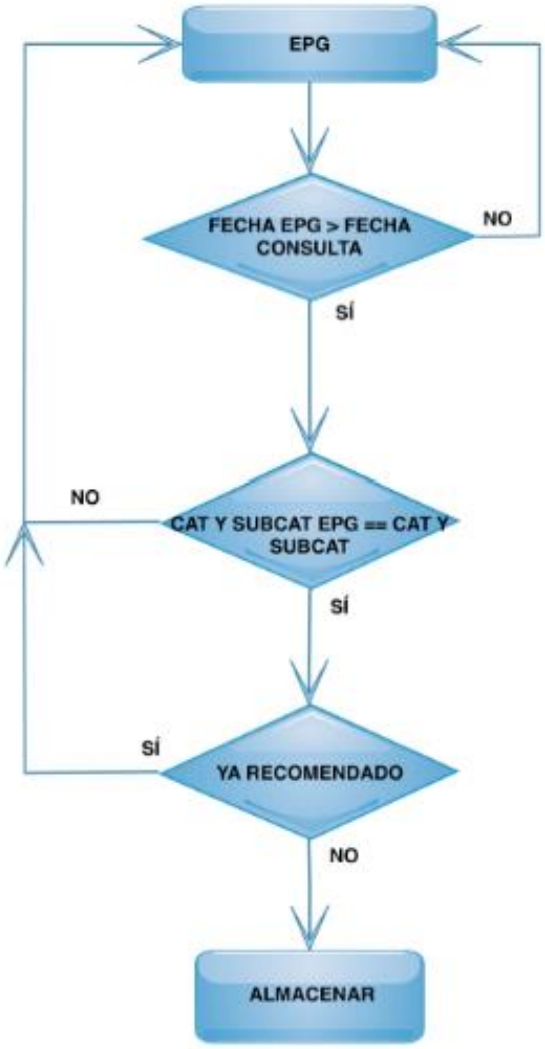

Figura 29. Diagrama de flujo de búsqueda de una recomendación

En la clase EpgHandler se realiza la búsqueda. Sabiendo que es una búsqueda de segundo tipo, se entiende que se quieren buscar recomendaciones. Lo primero que se realiza al leer un programa nuevo en la EPG es comparar la fecha de emisión. Si ésta es posterior a la fecha de la consulta pasa a comprobarse el contenido del programa. Posteriormente si la categoría y la subcategoría del evento son las mismas que las de la búsqueda se supone que el contenido es potencialmente del gusto del usuario y se añade a un objeto de tipo Event. Es posible que los resultados de esta búsqueda coincidan con los resultados de la búsqueda de programas concretos. Para no ofrecer dos veces los mismos resultados al terminar esta búsqueda se comprueba si es un programa que ya se va a recomendar por ser un programa periódico, de ser así, no almacena el programa en la base de datos de recomendaciones y pasaría al siguiente elemento en la búsqueda.

La búsqueda de recomendaciones se basa en la coincidencia de programas con la misma asignación de subcategorías, no obstante para personalizar un poco más el resultado, se da la opción al usuario de filtrar las recomendaciones únicamente al horario en el que el mismo suele ver la

televisión. Esta opción no modifica la búsqueda en sí, sino que si el usuario la ha activado, al mostrar las recomendaciones se aplicará un filtro que elimina aquellas que están fuera del marco horario donde se utiliza la aplicación. El marco donde el usuario ve la televisión queda definido por la media, que es calculada cuando hay suficientes muestras. Se ha establecido un mínimo de cinco muestras para poder activar esta opción, cuantas más muestras haya, mejor será el resultado. A esa hora media se le resta una hora por detrás y se le suma una por delante, quedando únicamente disponibles los programas que estén en ese rango. De esta manera, al usuario sólo se le recomendarán programas efectivos, es decir, que tengan más posibilidad de ser vistos.

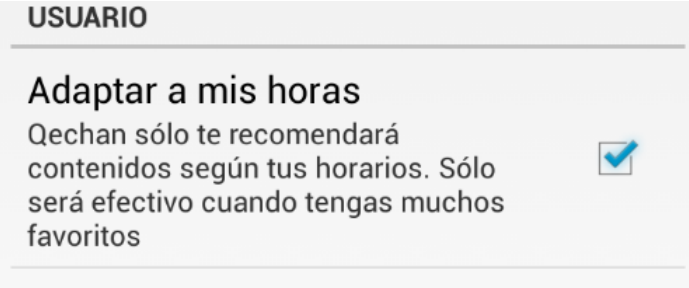

Figura 30. Opción para adaptarse al usuario

4.3. Ejecución de la inteligencia

Ya hemos visto cómo se ha diseñado la lógica de la aplicación y cómo se ha implementado, o al menos, a grandes rasgos y en los aspectos más importantes de la misma. En este punto se trata de explicar al lector cómo, cuándo y dónde se ejecuta la parte de la aplicación que el usuario no ve y que ha sido diseñada como base del trabajo fin de grado. Es decir, en este punto se explicarán los detalles que se han debido tener en cuenta para poder ejecutar las búsquedas en la EPG.

4.3.1. Ejecuciones en segundo plano mientras se usa la aplicación

La lectura de un archivo siempre es una tarea pesada y que requiere recursos. En esta aplicación se necesita leer un archivo XML durante diferentes momentos y, dependiendo del usuario, quizás muchas veces, por lo que no conviene que la lectura de la EPG se realice al mismo tiempo que se ejecuta la aplicación. De hecho, Android desde la versión 3.0, no permite la lectura de archivos XML en el mismo hilo[10] de ejecución que la aplicación, en realidad no se permite que se realice una tarea de larga duración de manera síncrona, por tanto hay que utilizar hilos asíncronos que se pueden implementar gracias a Java. Estos hilos o tareas asíncronas se ejecutan en segundo plano de manera que los recursos que consumen son menores porque permiten que la aplicación siga ejecutando el hilo principal sin problemas. Así pues, cuando la

aplicación requiere leer de la EPG el programa actual que el usuario está viendo, es necesario abrir una tarea asíncrona (*AsyncTask*) [34] para poder seguir manejando la interfaz gráfica mientras la aplicación busca en la EPG.

En el código se ha tenido que crear una clase dentro de la clase principal que es la encargada de activar el proceso de búsqueda de manera asíncrona. La clase *ReadEPGProcess* que para ser asíncrona debe heredar de *AsyncTask.* Dentro de esta clase podemos realizar tareas antes de la ejecución, durante la ejecución y después de la misma. Esto se aprovecha para ejecutar el método de lectura de EPG durante la ejecución y el método de mostrar información al terminar la ejecución.

```
private class ReadEPGProcess extends AsyncTask<Void, Integer, Boolean>{
    private ProgressDialog pDialog = new ProgressDialog(QechanActivity.this);
    @Override
    protected Boolean doInBackground(Void... params) {
        getCurrentEPG(canalActual.getName()); //llamamos al metodo para
                                              //parsear la EPG (XML)
        return true;
    }
    @Override
    protected void onProgressUpdate(Integer... values) {
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        pDialog.setMessage("Obteniendo información...");
        pDialog.setCancelable(true);
        pDialog.show();

    }
    @Override
    protected void onPostExecute(Boolean result) {
        pDialog.dismiss();
        if(!errorEpgParseSax)
            showEPG();
        else{
            Log.i("No se muestra", "EPG");
        }

    }
    @Override
    protected void onCancelled() {
         Toast.makeText(QechanActivity.this, "Búsqueda cancelada",
                 Toast.LENGTH_SHORT).show();
    }

}
```

Figura 31. Clase para definir la tarea asíncrona

Para usarla es necesario crear una instancia de la misma, que se crea cuando el usuario cambia el canal. En ese momento es necesario leer la EPG para mostrar el programa actual, y cuando se ejecuta la instancia creada se pasa a analizar el documento XML. Como se ve en el código, cuando la tarea

finaliza se pueden ejecutar las tareas necesarias. En el momento que finaliza la búsqueda del programa actual son necesarias dos cosas: mostrar la información y comenzar a aprender el contenido.

Para iniciar la tarea de aprendizaje se necesita otra tarea asíncrona debido a que es de larga duración, pero no es conveniente enlazar dos hilos asíncronos porque perdería la utilidad, por tanto al terminar el proceso de leer la EPG se indica que se muestre la misma con el método *showEPG*, y tras realizar las funciones para mostrar la información, dentro de ese mismo método se inicia la tarea que se encarga del aprendizaje.

Como se ha comentado en el capítulo 3 de la presente memoria, el método de aprendizaje va ligado a una operación de tiempo. Se realiza una comprobación cada minuto consultando si el usuario ha cambiado de canal, si han pasado 5 minutos y el usuario no ha cambiado de canal la aplicación interpreta que lo que están emitiendo en ese momento es algo que gusta al espectador, por tanto, pasa a almacenarlo en la base de datos. Para realizar la comprobación del canal cada sesenta segundos se utiliza otra característica de los hilos asíncronos muy útil; ponerlos a dormir. Cuando se pone un hilo a dormir, es decir, cuando se detiene momentáneamente, se pueden realizar otras acciones (como por ejemplo que el usuario cambie el canal), pero transcurrido el tiempo durante el cual el hilo está pausado se retoma la actividad del mismo en el mismo instante. En ese instante se comprueba si el usuario ha cambiado de canal, y de no haberlo hecho se realiza una comprobación de la hora. Si han transcurrido 5 minutos desde que el usuario hizo el último cambio de canal es cuando la aplicación entiende que debe almacenar lo leído por la EPG.

```
protected Boolean doInBackground(Void... params) {

    while(channel==canalActual.getName()){
        try {
            Thread.sleep(60000);//cada 60 segundos comprobamos si
                                //seguimos en el mismo canal
            calendar2 = Calendar.getInstance();
            fiveMinutes = tiempoTranscurrido(calendar1, calendar2);
            //fiveMinutes=true;
            if(fiveMinutes){
                /*si han pasado mas de 5 minutos lo almaceno
                para que no actualice los visionados de un mismo programa
                comrpobamos que la fecha en la que estamos viendo la TV
                es diferente al dia anterior*/
                repeat=isRepeat(program_name);
                if(repeat==0){
                    //0 contenido nuevo
                    //si es un contenido nuevo creamos el nuevo evento en la BD
                    newContent(user, program_name, program_nameorg, date,
                            canalActual.getName(), description, tipo1, tipo2,
                            viewing, proposed, recomended, hour);
                    channel=""; //para que termine de comprobar
                }
                return true;

            }
        } catch (InterruptedException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
}
    return true;
}
```

Figura 32. Método de aprendizaje

Cuando se ha almacenado el contenido se deja de comprobar el canal y la tarea asíncrona se termina.

4.3.2. Ejecuciones mientras la aplicación está cerrada

Hemos visto cómo se realizan las tareas en segundo plano mientras la aplicación se está ejecutando, a la vez que el usuario maneja la interfaz gráfica. Para que lo realizado por esas tareas, sobre todo por la tarea de aprendizaje, tenga sentido, es necesario trabajar cuando la aplicación no se está ejecutando, o al menos no necesitar que se ejecute para realizar estas tareas, y por ello es necesario el diseño de una solución que acometa este problema.

La tarea que debe realizarse sin que la aplicación esté en primer plano es la que requiere todas las búsquedas que generan las posibles recomendaciones o recordatorios. En un primer momento se intenta diseñar este proceso fuera del teléfono móvil, realizar las búsquedas en un servidor y que genere notificaciones *push* [35] para que sea el mismo el que realice las peticiones al cliente (la aplicación), pero se rechaza principalmente por no

disponer de la infraestructura necesaria y con el fin de fomentar la búsqueda de una solución que se implemente en el teléfono utilizando las herramientas disponibles en Android y seguir centrados en una aplicación ejecutada en el terminal móvil.

Para diseñar una solución que se desarrolle en el móvil se medita crear una programación de tareas. Programar una tarea que se lance en un momento concreto y que únicamente necesite, para lanzarse, que el móvil esté conectado, aunque no con la aplicación abierta, algo similar a las tareas que se programan con *cron*[11] en los sistemas operativos Unix. La solución dada consiste en un sistema de alarmas que se programan a una hora concreta, pero cuando llega la hora en la que está programada en vez de hacer sonar un tono musical, se lanza una tarea.

Cuando el usuario sale de la aplicación por completo, no la deja en segundo plano sino que la va a dejar de usar, se genera una alarma que se activará a las tres de la madrugada del día siguiente. El motivo de elegir esta hora es simple. De madrugada el uso del móvil suele ser menor por lo que si la tarea de búsqueda se ejecuta en un smartphone con poca capacidad el usuario se verá menos afectado por el posible consumo de recursos. Hay otro motivo para realizar la búsqueda de madrugada, y es que la actualización del archivo de EPG se realiza también por la noche, por lo que de esta manera se buscará siempre sobre unos datos actualizados.

```
protected void onDestroy() {
    if (mAnymoteClientService != null) {
        mAnymoteClientService.detachClientListener(this);
        unbindService(mConnection);
        Toast.makeText(QechanActivity.this, "Aplicación cerrada. " +
                "Conexión con Goggle Tv perdida",Toast.LENGTH_SHORT).show();
    }
    Intent intent = new Intent(ctx, AlarmReceiver.class);
    String tiempo = preferences.getString("antelacion", "30");
    intent.putExtra("tiempo", tiempo);
    am = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
    Calendar alarmTime = Calendar.getInstance();
    alarmTime.add(Calendar.DAY_OF_MONTH, 1);
    alarmTime.set(Calendar.HOUR_OF_DAY, 3);
    alarmTime.set(Calendar.MINUTE, 0);
    //alarmTime.set(Calendar.SECOND, 3);
    PendingIntent pendingIntent = PendingIntent.getBroadcast(ctx, 0,intent,
            PendingIntent.FLAG_ONE_SHOT);
    Log.i("onDestroy","Se lanzará búsuqeda a las "+alarmTime.getTime());
    am.set(AlarmManager.RTC_WAKEUP,alarmTime.getTimeInMillis(), pendingIntent);
    finish();
    super.onDestroy();
}
```

Figura 33. Código para programar la tarea de búsqueda

La tarea que se ejecuta está definida en la clase *AlarmReceiver* donde se lanza un servicio de búsqueda instanciando a la clase *SearchService*. Esta clase define también una tarea asíncrona, como las clases usadas para la lectura de la EPG mientras se ejecuta la aplicación. En ella se realizan los tipos de búsquedas definidos anteriormente. Para realizar la búsqueda de una

manera más ordenada se aprovechan los métodos definidos para antes, durante y después de la ejecución. Así en el método *onPreExecute* se analizan los datos almacenados en la base de datos que contiene los gustos del usuario y cuyo contenido son los que generan las búsquedas. De este análisis se obtienen los programas que han sido vistos más de una vez, las categorías que son más repetidas… Este análisis permite que en el método *doInBackGround* se instancie a la clase que analiza la EPG (EpgParserSax) y según los parámetros poder buscar lo que interese en el archivo XMLTV.

En esta tarea, se da importancia también al método *onPostExecute* ya que es donde se ha definido la creación de notificaciones. Para crearlas se analiza el contenido de una tercera base de datos que contiene los programas que van a ser notificados al usuario. De manera automática pasan a esta base de datos los programas que son periódicos, pero el usuario, de manera manual, puede elegir que alguno de los programas que le han sido recomendados también le sean notificados. En este caso se copia el contenido de la base de datos que contiene las recomendaciones a la que contiene las notificaciones. Por defecto, las notificaciones se lanzan con una antelación de treinta minutos sobre la hora de inicio, pero el usuario puede modificar este tiempo a su gusto.

4.4. Esquema de clases

En este apartado se presenta un esquemático de la relación que existe entre las clases Java. Para llevar a cabo la aplicación se han ido añadiendo clases según las exigencias. En total hay diseñadas veintisiete clases Java independientes y dos clases definidas dentro de otra.

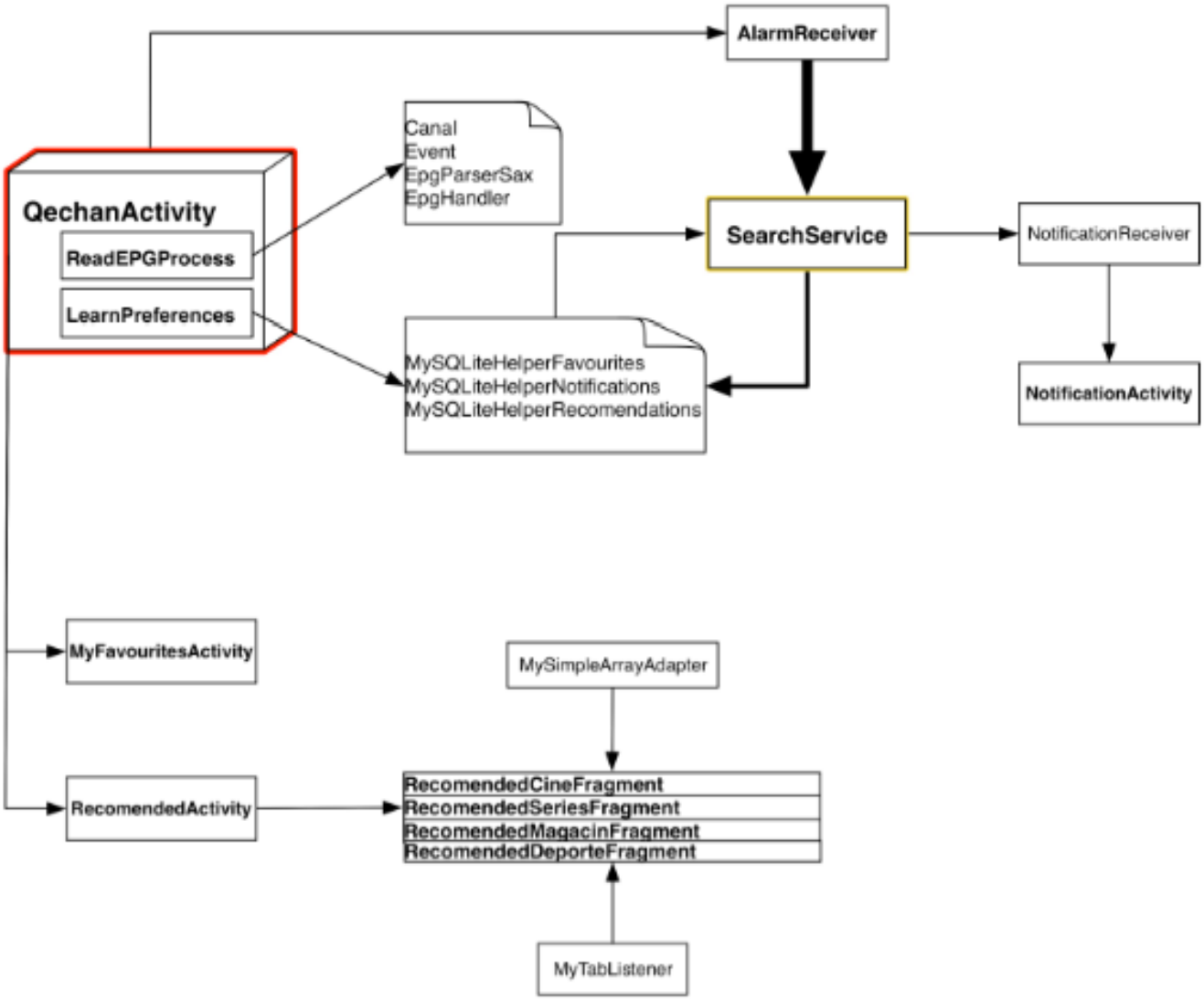

Figura 34. Esquema de clases (I)

En este esquema están contenidas las clases que se utilizan para la funcionalidad de manera estricta. Destacar que todas las clases que implementan una interfaz gráfica y heredan de la clase *Activity*, llevan el nombre Activity en su título. Al igual que los *Fragment*, trabajar con fragmentos es una tónica muy actual en Android y para diseñar una ventana con pestañas es lo más eficiente.

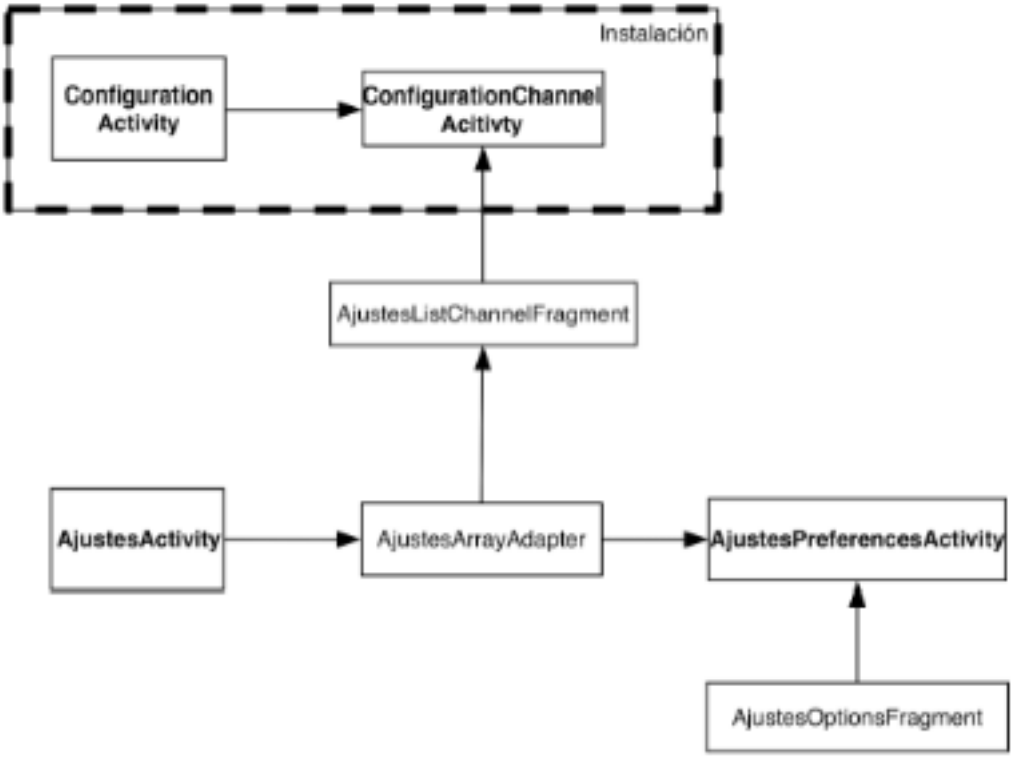

Figura 35. Esquema de clases (II)

En éste lo que vemos son las clases que hacen referencia a la parte de la aplicación que está por detrás de la funcionalidad, las opciones. Aunque la aplicación no podría funcionar sin rellenar las opciones que se presentan la primera vez que se arranca la aplicación, no es una parte del código tan importante. El resto de clases se utilizan para afinar aún más la aplicación al gusto del usuario y para modificar las opciones iniciales por si sufren cambios.

Lo que se intenta mostrar con estos esquemas es la dependencia de unas clases con otras, pues, pese a estar definidas de manera independiente, todas son necesarias. Una fragmentación debe hacerse con cuidado de no resultar compleja, pero a su vez facilita el trabajo a la hora de actualizar la aplicación. Al final, desde la clase principal de la aplicación, QechanActivity.java, todo está conectado.

A estos dos esquemáticos de relación entre clases se debe añadir también todas las clases de la AnymoteLibrary, que permite la comunicación con Google TV. Sin embargo al ser una librearía y teniendo acceso libre se ha preferido adjuntar una referencia para poder visitarla. [36]

Capítulo 5

5. Pruebas de rendimiento

Las aplicaciones móviles requieren una fase detallada de pruebas de rendimiento. En los grandes proyectos nos encontramos con gente que ha sido ajena al desarrollo del proyecto y que se dedica a utilizar la aplicación, con el fin de detectar fallos en condiciones de estrés o fallos de manejo. Este apartado es tan importante como cualquier otro porque es el último paso antes de lanzar una aplicación al mercado.

En este proyecto no se ha podido contar con un grupo de gente con el que poder hacer pruebas, simplemente se han supuesto diferentes casos de uso y se ha intentado adelantar el comportamiento del usuario para que la aplicación sepa responder a las diferentes situaciones sin provocar errores. Las pruebas de rendimiento han hecho que se modifiquen determinados aspectos, por ello no es algo que se haya dejado para el final, si no que ha sido un proceso paralelo a lo largo del proyecto.

Las pruebas han demostrado que es necesario mostrar información al usuario y también cómo desarrollar ciertas implementaciones, como las búsquedas en la EPG debido a la dificultad que conllevan.

5.1. Comunicación con el usuario

Según se ha ido probando la aplicación se han generado diferentes situaciones que, sin la debida información, pueden llevar al usuario a pensar que no se está utilizando bien la aplicación o incluso, que no funciona. Al tratarse de una aplicación móvil, se demanda un uso sencillo y eso se ha tenido en cuenta en la interfaz gráfica, pero es necesario transmitir información que no se puede comunicar mediante un manual de instrucciones, por ello se necesitan elementos efímeros para dar información al usuario.

A continuación se muestran diferentes casos en los que es posible que el usuario necesite algo de información para manejar la aplicación de manera correcta.

- Conexión a Internet: como ya se ha comentado la aplicación requiere de conexión a la misma red Wi-Fi que Google TV para poder manejarlo y como mínimo requiere de una conexión a Internet a través de conexión de datos para poder consultar los archivos de EPG. Si no se dispone de ella, al usar la aplicación se lanza un mensaje.

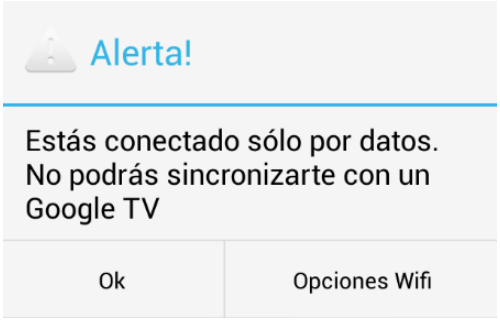

Figura 36. Ejemplo de alerta sobre conexión a Internet

- Datos requeridos: si el usuario no completa los datos mínimos sobre su información la aplicación no podrá funcionar correctamente. Como mínimo la aplicación necesita la localización de cada canal en el descodificador TDT del usuario.

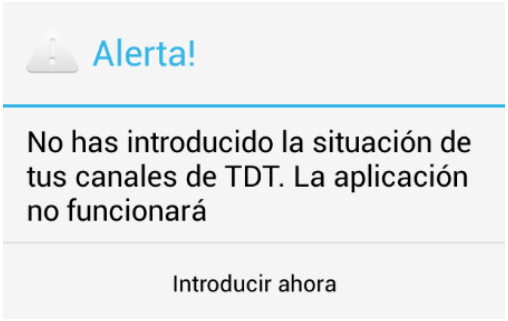

Figura 37. Aviso al usuario sobre el requerimiento de datos

- Avisos: cuando el usuario va a cerrar la aplicación se procede a avisarle para evitar que cierre la misma de manera no intencionada, ya que esto implica, de existir, perder la conexión con Google TV.

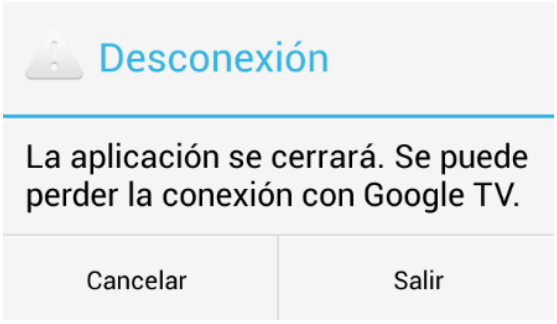

Figura 38. Diálogo mostrado antes de salir de la aplicación

- Consulta de EPG: las consultas a la EPG se realizan de manera asíncrona y se han sometido a un estrés suficiente como para asegurar su buen funcionamiento. Se ha cambiado de canal antes de que haya terminado la búsqueda para comprobar que no existen problemas a la hora de cancelaciones. Para intentar que el usuario no se impaciente, mientras se realiza la búsqueda aparece un *spinner*[12] indicando que se requiere un tiempo mínimo para mostrar la información.

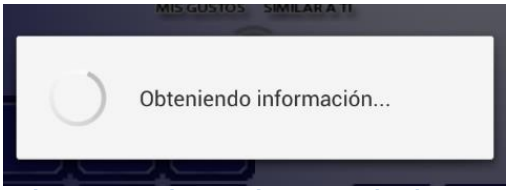

Figura 39. Spinner mientras se lee la EPG

Como se ve se han usado diálogos en la mayoría de ocasiones. Es un recurso rápido y sencillo para aportar la información necesaria al usuario.

5.2. Aplicación en segundo plano

Si un usuario manda a segundo plano la aplicación se considera que la abrirá de nuevo, en un corto periodo de tiempo. Por ello, cuando esto sucede, se mantiene el estado que tuviera la aplicación. Es decir seguirá teniendo en cuenta que el canal no ha cambiado y que sigue viéndolo. Por tanto si el usuario no quiere que la aplicación aprenda datos que no son ciertos debe cerrar la aplicación. Es algo que no todo el mundo puede entender de manera intuitiva, por tanto se explica en el apartado de ayuda de la aplicación.

A este apartado se accede mediante las opciones de la aplicación y aparece una pantalla como la siguiente, donde se explican el tipo de cosas menos intuitivas.

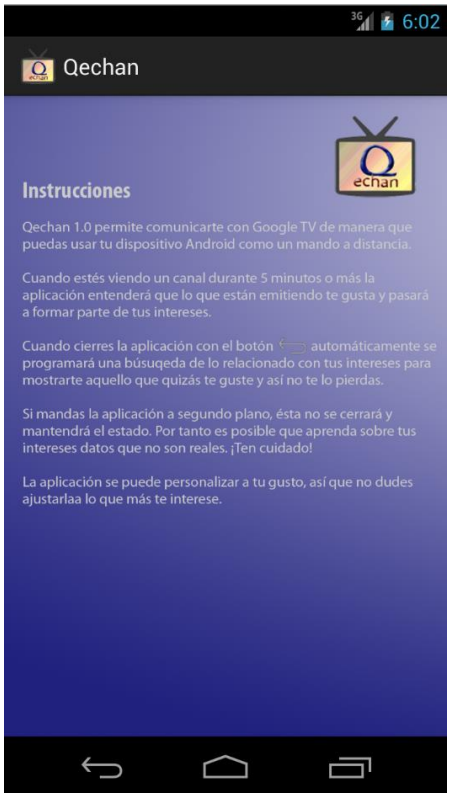

Figura 40. Pantalla de instrucciones

5.3. Aplicación cerrada

Cuando la aplicación se cierra se termina la comunicación con Google TV, de haberse iniciado. También se programa una alarma para realizar búsquedas y esto o curre cada vez que la aplicación es cerrada. Las alarmas se sobrescriben, porque siempre saltan a la misma hora. Es posible que en algún caso se realice una búsqueda sin datos nuevos porque la aplicación no ha aprendido nada nuevo con lo que buscar, pero la EPG cambia a diario por lo que la información a comparar siempre es nueva y por tanto siempre existe la posibilidad de encontrar resultados satisfactorios, aunque los datos con los que comparar no hayan variado.

Se decide fijar una hora en la que en la mayoría de casos el móvil no se está usando porque la tarea de búsqueda puede hacer que otras tareas se ralenticen si se están ejecutando al mismo tiempo ya que consume una cantidad de recursos moderada y si la máquina donde se ejecuta no dispone de mucha memoria puede verse afectada.

Capítulo 6

6. Líneas futuras

El mundo de las aplicaciones móviles tiene un potencial grandísimo. Como aplicación móvil, este proyecto, también. Los requisitos planteados en un primer momento se han cumplido, en algún caso incluso, superando las expectativas iniciales sobre los resultados, aunque resulte innegable que se puede ampliar en diferentes puntos. También se puede mejorar lo ya hecho a las nuevas actualizaciones que presente Google sobre el sistema operativo Android.

6.1. Mejoras de implementación

Como se ha mencionado, los requisitos iniciales se han cumplido y la funcionalidad de la aplicación es total. Pese a ello, hay determinados puntos que se pueden pulir y conseguirían un resultado más óptimo.

6.1.1. Carga de canales

La información acerca de la situación de los canales en el descodificador es algo que ha condicionado la aplicación. Por el momento no parece que Google vaya a dar ningún paso en este sentido, pero para que la lista de canales de los cuales se pueda elegir no sea estática y sea dinámica, se ha pensado en que el usuario introduzca su localización o incluso se pueda localizar mediante el uso de Internet o GPS y así cargar una lista de canales de TDT que tenga las particularidades de cada provincia, como son los canales autonómicos, locales o las variaciones de los canales nacionales.

6.1.2. Búsquedas EPG

Las búsquedas en la EPG podrían diseñarse según el resultado que se quiera conseguir. De este modo, en vez de reutilizar el método analítico, se podría implementar un método de almacenamiento y análisis posterior que mejoraría los resultados aunque no el rendimiento. Incluso podrían mantenerse ambos métodos y establecer un algoritmo comparativo para decantarse por unos resultados u otros.

A su vez, se puede mejorar el algoritmo de búsqueda y establecer unos parámetros determinados con el fin de evitar el problema del arranque en frío en las recomendaciones. Se podría añadir un menú de opciones al instalar la aplicación que pida al usuario una pequeña pincelada sobre sus gustos y así, antes de aprender el contenido concreto que ve, poder recomendar eventos de manera pseudoaleatoria.

6.1.3. Interfaz gráfica

La interfaz gráfica se puede mejorar, no para hacerla más llamativa sino más funcional todavía. Si se implementan nuevas opciones se pueden aprovechar más los espacios para añadir nuevos botones.

Disponer de una segunda interfaz en caso de que el teléfono se oriente en sentido horizontal sería útil, habilitando más espacio, y ampliando la posibilidad de disponer, por ejemplo, de un mando a distancia más completo que permita otras funcionalidades además de cambiar de canal y subir y bajar el volumen.

Otro aspecto a mejorar de la interfaz gráfica podría ser el espacio reservado a mostrar la información de la EPG. Si la aplicación se ejecuta en una pantalla más grande o en sentido horizontal, este espacio podría variar el tamaño y mostrar más información. Este aspecto actualmente se soluciona pinchando en el recuadro reservado para la información, pero podría mejorarse.

También es conveniente tener en cuenta que se debe dotar la aplicación de multilenguaje. En este proyecto sólo se ha tenido en cuenta el castellano, pero Google TV es un producto estadounidense y con comercialización en aquel país, por lo que la actualización más interesante de la interfaz es dotarla de información también en inglés.

Si se dispusiera de más tiempo se podría diseñar otra opción para cambiar el canal que no fuera el mando a distancia gráfico, y se podría así habilitar una opción para usar el movimiento del teléfono para cambiar el canal. Si un usuario mueve su dispositivo de izquierda a derecha o de derecha a izquierda el canal aumentará o disminuirá. Esta idea ya fue planteada en el desarrollo, pero se descartó y se dejó como una opción futura debido al gran trabajo que conlleva y porque no es una funcionalidad principal.

6.1.4. Categorías correctas

Tal y como se indicó no todos los contenidos aparecen bien etiquetados en la EPG. Para solucionar este problema se ha ideado una posible solución donde el usuario tiene mucho que decir. Si el usuario considera que un contenido está asociado a una categoría pero en la EPG no aparece así, sería conveniente forzar a la aplicación para que tuviera en cuenta la opinión del usuario frente a lo que viene en la EPG. Por tanto, se piensa en introducir una opción para canales temáticos, cuyo contenido siempre gira en torno a un tipo de contenido, para que el usuario indique a la aplicación la categoría que uno piensa que es. Es decir, si hay un canal temático en la TDT española como es Boing, el usuario puede indicar que todos sus contenidos son series en el caso que dentro de la guía electrónica de programa vengan etiquetados como magacines.

6.2. Sociabilidad

A lo largo de la memoria se ha comentado que el algoritmo diseñado para las recomendaciones se basa en resultados personales y no hay otra

manera de obtener resultados que no sea mediante el propio usuario. El siguiente paso para mejorar la aplicación y casi obligatorio, es hacer de la aplicación una red social o, en su defecto, un complemento de las mismas.

En esta primera versión se ha tenido en cuenta la sociabilidad sólo para compartir información. Un usuario puede indicar a sus amigos lo que está viendo pulsando un botón y seleccionando la aplicación de su móvil donde la quiere compartir (Facebook, Twitter, Whatsapp…) pero no va más allá.

Para sociabilizar la aplicación se crearía un registro donde el usuario, si así lo deseara, registrara su aplicación, es decir, si el usuario quiere vivir una experiencia social simplemente debe registrarse como usuario y a partir de aquí la información que se guarda en el teléfono sobre dicho usuario, debería almacenarse en un servidor al que sólo el administrador de la aplicación tiene acceso. De esta manera el algoritmo de recomendación cambiaría. Se habilitaría una opción para agregar a otros usuarios que estén en esa base de datos, los usuarios podrán registrarse con un nombre, con su Twitter o con su Facebook de modo que es más fácil localizar a nuestros amigos. El nuevo algoritmo tendría dos o tres variantes. Una simple sería recomendar lo más visto entre todos los usuarios, que normalmente será algo que está muy de moda. También se puede implementar una búsqueda común entre usuarios y de entre dos listas de categorías recomendar a un usuario algún contenido que no esté en su lista pero sí en la de un amigo suyo.

Las ventajas que supone sociabilizar la aplicación son muchas y variadas, y la complejidad no es demasiada pero sí supone mucho tiempo, dedicación y una pequeña infraestructura donde poder almacenar los datos.

6.3. Migración

Migrar la aplicación Qechan a otro sistema operativo no tiene mucho sentido hoy por hoy. Bien es cierto que existe una versión de Anymote para sistemas iOS de Apple, pero la mayoría de usuarios que se decantan por tener un Google TV tienen un smartphone o tableta Android, por lo que es más rentable diseñarla sólo para Android y lograr un buen resultado. Hablar de otros sistemas como Windows Phone no tiene sentido porque ni siquiera existe la posibilidad de manejar Google TV desde este sistema operativo. Por otro lado, tampoco cabe la opción de desarrollarla en un sistema operativo Bada, propio de Samsung, ya que es el sistema operativo de sus Smart TV y por tanto vienen con aplicaciones propias que son compatibles para teléfonos Bada y Android y el usuario de un televisor de este tipo no necesita pues Google TV.

Capítulo 7

7. Planificación y presupuesto

7.1. Diagrama Gantt

En este apartado se detalla la duración del proyecto. Aproximadamente nueve meses durante los cuales el desarrollo del mismo se ha compaginado con una jornada laboral a tiempo completo y estudios académicos.

A continuación se muestra la lista de tareas más importantes del proyecto de manera cronológica, también se muestra utilizando el modelo de Gantt.

Tabla 1. Tareas del trabajo fin de grado

Tarea	Fecha Inicio	Fecha Fin
Documentación sobre Google TV	03/12/12	19/12/12
Documentación para desarrollar para Google TV	03/12/12	17/12/12
Definición de la idea	18/12/12	19/12/12
Compra de STB Google TV	24/12/12	18/01/13
Acondicionamiento de los equipos	24/12/12	26/12/12
Aplicación de prueba	08/01/13	14/01/13
Pruebas de conectividad con GTV	21/01/13	23/01/13
Compra de STB Google TV compatible con TDT	24/01/13	13/02/13
Diseño de la aplicación	25/01/13	30/01/13
Diseño de GUI	28/01/13	29/01/13
Prueba con GTV	14/02/13	15/02/13
Adquisición EPG	18/02/13	29/03/13
Lectura XML	20/02/13	04/04/13
Búsquedas en EPG	05/04/13	02/05/13
Algoritmo de aprendizaje	03/05/13	13/06/13
Algoritmo de recomendación	17/05/13	28/06/13
Notificaciones y Opciones	01/07/13	05/07/13
Memoria	08/07/13	26/08/13
Presentación	27/08/13	04/09/13

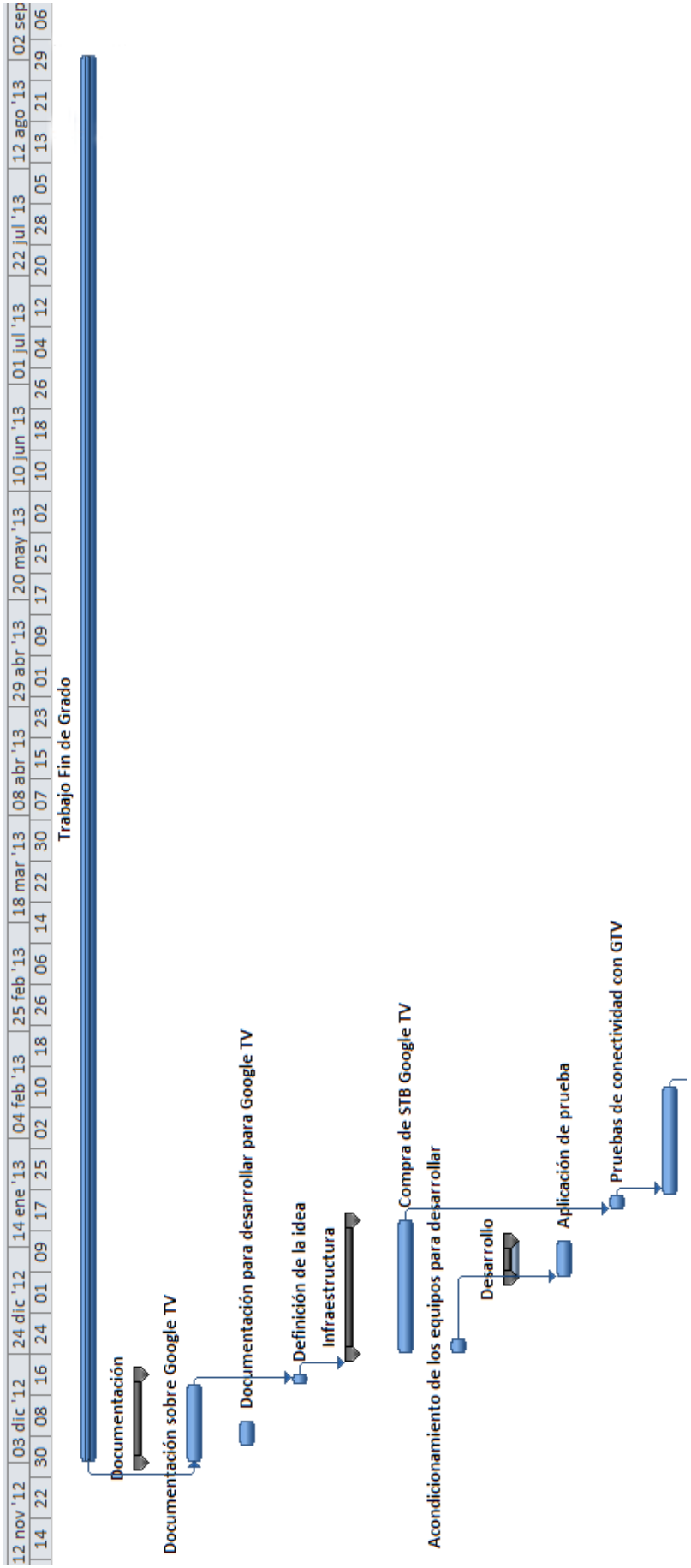

Figura 41. Diagrama de Gantt (I)

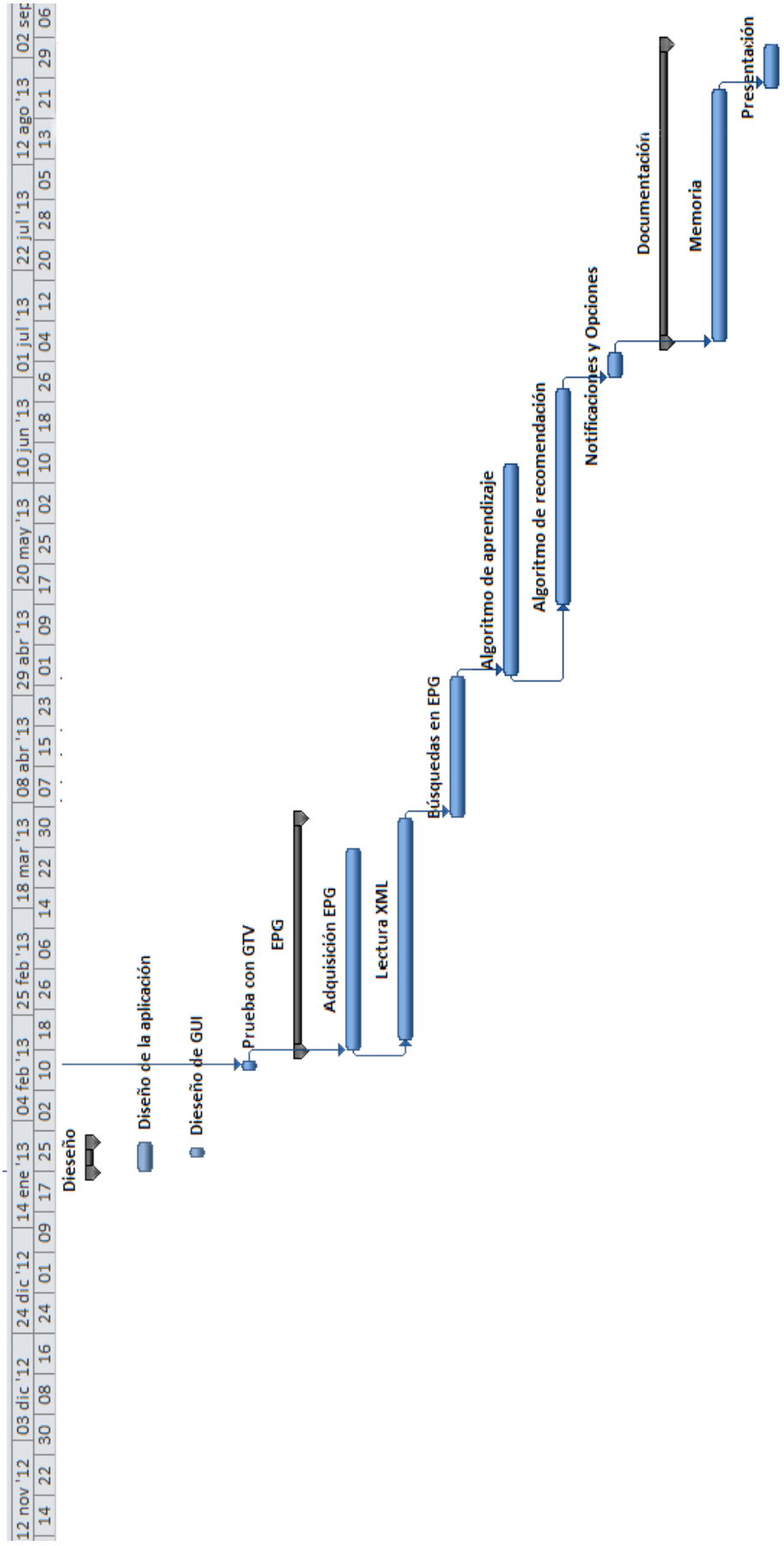

Figura 42. Diagrama de Gantt (II)

7.2. Presupuesto

Con la duración del proyecto y sus tareas detalladas, así como las herramientas necesarias para llevarlo a cabo, se desglosan los costes ligados a la realización de todas las etapas del proyecto. Para desglosar los costes se ha utilizado la plantilla facilitada en los recursos del trabajo fin de grado en la Universidad Carlos III de Madrid. [37]

Tabla 2. Presupuesto. Personal

Apellidos y Nombre	Categoría	Dedicación (Hombre MES)	Coste hombre mes (Euros)	Coste
Martínez Tejero, Diego	Ingeniero	9	2.694,39 €	24.249,51€

En cuanto a equipos necesarios y software de los mismos se detallan los gastos en la Tabla 3. Se ha tenido en cuenta un uso de cada elemento de un 100% y un precio de los equipos sin IVA. Para calcular el coste imputable se ha calculado utilizando la siguiente fórmula.

$$\frac{A}{B} \text{xCxD}$$

Figura 43. Fórmula del coste imputable

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de deprecación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (Habitualmente 100%)

Tabla 3. Presupuesto. Material

Concepto	Coste (Euros)	% Uso dedicado proyecto	Dedicación (meses)	Precio de deprecación	Coste imputable
Macbook Pro 8,1	1.180,99 €	100	9	60	177,15 €
PC	388,43 €	100	5	60	32,37 €
Xperia Neo V	185,95 €	100	4	60	12,40 €
Asus Nexus 7	189,25 €	100	2	60	6,31 €
Vizio Google TV	82,63 €	100	1	60	1,38 €
Hiense Pulse Google TV	82,63 €	100	4	60	5,51 €
TDT HDMI Sunstech	33,05 €	100	4	60	2,20 €
Monitor LG IPS224 22''	131,40 €	100	9	60	19,71 €
Apple OSX 10.8.5	0,00 €	100	9	60	0,00 €
Windows 7 Ultimate	148,76 €	100	5	60	12,40 €
Ubuntu 12.04 LTS	0,00 €	100	5	60	0,00 €
Android 4.0	0,00 €	100	4	60	0,00 €
Eclipse Helios 3.6.2	0,00 €	100	9	60	0,00 €
ADT Plugin 22.0.1	0,00 €	100	9	60	0,00 €
Adobe Photoshop CS5	866,94 €	100	6	60	86,69 €
WebGrab+Plus V.1.1.1	0,00 €	100	5	60	0,00 €
				TOTAL	356,1125

El coste del proyecto total se refleja en la Tabla 4, donde se han tenido en cuenta unos costes indirectos del 20%, derivados de posibles dietas, desplazamientos y demás costes no previsibles. Además se ha imputado un 21% de IVA teniendo en cuenta que el proyecto se lleva a cabo en España.

Tabla 4. Presupuesto total

Concepto	Importe
Costes de personal	24.249,51 €
Coste de material	356,11 €
Costes indirectos	2.765,61 €
Coste sin IVA	27.371,23 €
TOTAL	33.119,19 €

En total, el coste del proyecto llega a treinta y tres mil ciento diecinueve euros con diecinueve céntimos.

Capítulo 8

8. Conclusiones

Una vez se ha concluido el proyecto y la redacción de la memoria se intenta hacer un balance del proceso para poder hacer una valoración y también diferentes reflexiones personales sobre el trabajo.

El proyecto ha finalizado con los objetivos iniciales cumplidos. A la hora de usar la aplicación se han ido puliendo diversos fallos para tener cada vez una versión estable. Es necesario decir que, aunque se dé por concluido el proyecto, tal y como se ha comentado, tiene mucho potencial por lo que sería interesante la marca de otros objetivos y seguir el desarrollo hacía otras vías. Me gustaría destacar la valoración positiva sobre la simplicidad de uso de la aplicación, ya que se realiza de manera intuitiva y ha sido valorado por diferentes usuarios ajenos al desarrollo del proyecto.

Durante los nueve meses que ha durado el proyecto se ha pasado por diferentes sensaciones personales. Afrontar un proyecto de estas características era algo completamente nuevo y ha supuesto un desafío. Los objetivos planteados para llevar a cabo la aplicación se han cumplido, superando las primeras expectativas sobre los mismos, de modo que eso ha hecho que se haya conseguido terminar la aplicación de manera exitosa. Haber desarrollado el proyecto en Android ha permitido asentar los conocimientos sobre la plataforma que adquirí durante la carrera, así como conocimientos generales de la programación que ayudan a la hora de plantear soluciones a diferentes problemas, además de aprender nuevos recursos para implementar ideas en las aplicaciones móviles.

A nivel personal puedo destacar diferentes aspectos. Haber realizado un proyecto fin de grado ligado con el ámbito televisivo ha sido una ventaja. Siempre me he interesado por las tecnologías que rodean a la televisión y desarrollar una aplicación móvil que esté, aunque casi de manera indirecta, relacionada con ella, ha hecho que pueda entender mejor lo que un usuario quiere a la hora de ver los contenidos que se emiten.

Como final quiero destacar que ha supuesto un trabajo y un esfuerzo distinto a lo que estaba acostumbrado a realizar en las tareas académicas. El hecho de llevar un proyecto y ser casi independiente a la hora de tomar decisiones sobre el mismo y también de cómo llevarlas a cabo, te hace adquirir cierta experiencia y te prepara para proyectos mucho más importantes que se deben afrontar a lo largo de la carrera profesional.

Glosario

[1]*Add-on:* se traduce como complemento o extensión. En programación se entiende como un añadido necesario para programar en un entorno específico utilizando las mismas plataformas.

[2]*MAC:* son las siglas de *Media Access Control*, es un identificador de 48 bits que es único para cada tarjeta o dispositivo de red.

[3]*Stream:* flujo de datos.

[4]*Elementary stream:* es un contenedor de datos cuya cabecera se define dependiendo del códec. Contiene diferentes datos y siempre guardan la misma estructura.

[5]*Set top box (STB):* término inglés que se utiliza para referirse a los descodificadores de manera genérica. Habitualmente se utiliza como americanismo.

[6]*Layout:* se podría traducir como tapiz, porque tras definirlo se colocarán diferentes objetos visibles encima. Tiene la peculiaridad que en el diseño de webs o aplicaciones, los hay con formas predefinidas que sólo dejan colocar objetos en puntos determinados.

[7]*D-Pad:* se utiliza para nombrar al conjunto de cuatro botones de flechas que es habitual encontrar en los mandos a distancia.

[8]*Get y set:* en programación estos métodos son un recurso común y habitual para asignar valores a los atributos de cada clase y para consultarlos.

[9]*URL:* de sus siglas *Uniform Resource Locator* se utiliza para referirnos a la localización de contenido en Internet mediante una dirección.

[10]*Hilo:* en programación se entiende como una secuencia de tareas.

[11]*Cron:* administrador de tareas en segundo plano en los sistemas Unix.

[12]*Spinner:* elemento gráfico con forma circular que se suele usar para indicar que una tarea aún sigue en proceso.

Anexo I

Se adjunta un enlace electrónico para tener acceso al código de la aplicación de manera libre.

https://github.com/diemate/qechanTFG

Bibliografía

Lauren Darcey, Shane conder. *Programación Android 4*. Ediciones Anaya Multimedia S.A. 2012.
ISBN: 978-84-415-3194-9
[Última consulta mayo 2013]

James F. Kurose, Keith W.Ross. *Redes de computadoras. Un enfoque diferente.* Pearson Educación, S.A. 2010
ISBN: 978-84-7829-119-9
[Última consulta julio 2013]

[1]. TDT. Ministerio de industria, energía y turismo. [Última consulta junio 2013]
http://www.televisiondigital.es/TDT/Paginas/canales-tdt.aspx
[2]. The cocktail analysis. IV Estudio sobre Mobile Marketing de IAB Spain y The Cocktail Analysis**.** [Última consulta mayo 2013]
http://tcanalysis.com/blog/posts/iv-estudio-sobre-mobile-marketing-de-iab-spain-y-the-cocktail-analysis
[3]. Emulación de Google TV. Google Developers. [Última consulta enero 2013]
https://developers.google.com/tv/android/docs/gtv_emulator?hl=de-DE
[4]. Cnet. *Over half of smart TVs never go online*. [Última consulta junio 2013]
http://m.cnet.com.au/over-half-of-smart-tvs-never-go-online-339344442.htm?redir=1
[5]. *Televisión terrestre en España*. Wikipedia, la enciclopedia libre. [Última consulta junio 2013]
http://es.wikipedia.org/wiki/Televisión_terrestre_en_España
[6]. *Quiero TV*. Wikipedia, la enciclopedia libre. [Última consulta junio 2013]
http://es.wikipedia.org/wiki/Quiero_TV
[7]. Panorama audiovisual. *Cisco Connect 2013: mañana empieza aquí*. [Última consulta julio 2013]
http://www.panoramaaudiovisual.com/2013/05/09/cisco-connect-2013-manana-empieza-aqui/
[8]. *Web 2.0* Wikipedia, la enciclopedia libre. [Última consulta julio 2013]
http://es.wikipedia.org/wiki/Web_2.0
[9]. *IV Estudio anual redes sociales.* IAB Spain. [Última consulta julio 2013]
http://www.iabspain.net/wp-content/uploads/downloads/2013/01/IV-estudio-anual-RRSS_reducida.pdf
[10]. *Industry Leaders Announce Open Platform to Bring Web to TV*. Google. [Última consulta julio 2013]
http://googlepress.blogspot.com.es/2010/05/industry-leaders-announce-open-platform.html
[11]. *Search across TV, Web, and Apps*. Google. [Última consulta julio 2013]
http://googlepress.blogspot.com.es/2010/05/industry-leaders-announce-open-platform.html
[12]. *User Interface*. Google Developers. [Última consulta julio 2013]
https://developers.google.com/tv/android/docs/gtv_android
[13]. *Optimize your website for Google TV*. Google Developers. [Última consulta julio 2013]
https://developers.google.com/tv/web/

[14]. *Second-screen Applications*. Google Developers. [Última consulta julio 2013]
https://developers.google.com/tv/remote/
[15]. *Anymote Protocol*. Google Developers. [Última consulta julio 2013]
https://developers.google.com/tv/remote/docs/anymote
[16]. [Última consulta. [Última consulta julio 2013]
https://developers.google.com/protocol-buffers/docs/reference/java/com/google/protobuf/MessageLite
[17]. *Request Messages. Response Messages*. Google Developes. [Última consulta julio 2013]
https://developers.google.com/tv/remote/docs/anymote
[18]. Capítulo 8. 8.5 *Conexiones TCP Seguras SSL*. *Redes de computadoras Un enfoque descendente* James F. Kurose, Keith W. Ross. [Última consulta julio 2013]
[19]. Google TV Pairing Protocol. Google Developers. [Última consulta julio 2013]
https://developers.google.com/tv/remote/docs/pairing?hl=es
[20]. *Guía electrónica de programas*. Wikipedia, le enciclopedia libre. [Última consulta julio 2013]
http://es.wikipedia.org/wiki/Guía_electrónica_de_programas
[21]. *XMLTVFormat*. Wiki XMLTV. [[Última consulta julio 2013]
http://wiki.xmltv.org/index.php/XMLTVFormathttp://wiki.xmltv.org/index.php/XMLTVFormat
[22]. WebGrab + Plus. [Última consulta agosto 2013
http://webgrabplus.com
[23]. SincroguiaTV. [Última consulta agosto 2013]
http://www.sincroguia.tv
[24]. Sergio Manuel Galán Nieto. *Filtrado Colaborativo y Sistemas de Recomendación*. [Última consulta julio 2013]
http://www.upf.edu/hipertextnet/numero-2/recomendacion.html
[25]. *Filtro Colaborativo.* Wikipedia, la enciclopedia libre. [Última consulta julio 2013]
http://es.wikipedia.org/wiki/Filtrado_colaborativo
[26]. *Arranque en frio*. Wikipedia, la enciclopedia libre. [Última consulta julio 2013]
http://es.wikipedia.org/wiki/Arranque_en_frío
[27]. *Sistemas de recomendaciones: herramientas para el filtrado de información en Internet*. [Última consulta julio 2013]
http://www.upf.edu/hipertextnet/numero-2/recomendacion.html
[28]. Salvador Gómez Oliver. *Bases de datos en Android (I): Primeros pasos.* [Última consulta agosto 2013]
http://www.sgoliver.net/blog/?p=1611
[29].Google TV Remote, source code. [Última consulta agosto 2013]
https://code.google.com/p/google-tv-remote/source/browse/
[30].*SlidingDrawer*. Android Developers. [Última consulta agosto 2013]
http://developer.android.com/reference/android/widget/SlidingDrawer.html
[31].*XMLPullParser.* Android Developers. [Última consulta agosto 2013]
http://developer.android.com/reference/org/xmlpull/v1/XmlPullParser.html

[32]. Salvador Gómez Oliver. *Tratamiento de XML en Android (I):SAX.* [Última consulta agosto 2013]
http://www.sgoliver.net/blog/?p=1542
[33]. *DefaultHandler.* Android Developers.[Última consulta agosto 2013]
http://developer.android.com/reference/org/xml/sax/helpers/DefaultHandler.html
[34]. *AsyncTask.* Android Developers [Última consulta agosto 2013]
http://developer.android.com/reference/android/os/AsyncTask.html
[35]. *Tecnología Push.* Wikipedia, la enciclopedia libre. [Última consulta agosto 2013]
http://es.wikipedia.org/wiki/Tecnología_Push
[36]. *Google TV-android-samples. AnymoteLibrary.* Google TV. [Última consulra agosto 2013]
https://code.google.com/p/googletv-android-samples/source/browse/ -git%2FAnymoteLibrary
[37].Plantilla presupuesto TFG UC3M.
http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera